TXED Text Editor and Formatter

for Disc Systems

RESEARCH MACHINES

Reference Manual

TXED Version 4 Reference Manual

Release 3, September 1981

Copyright (c) 1981 by Research Machines Limited.

All rights reserved. Copies of this publication may be made by customers exclusively for their own use, but otherwise no part of it may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language without the prior written permission of Research Machines Ltd., Post Office Box 75, Mill Street, Oxford OX2 0BW, England. Tel. Oxford (0865) 49866.

The policy of Research Machines Limited is one of continuous development and improvement of its products and services, and the right is therefore reserved to revise this document or to make changes in the computer software it describes without notice. RML make every endeavour to ensure the accuracy of the contents of this document but do not accept liability for any error or omission.

The original labelled distribution disc is regarded as the only proof of purchase and must be produced in order to qualify for an update at a reduced rate. Keep it safe and always work from copies.

Additional copies of this publication may be ordered from Research Machines at the address above. Please ask for "TXED Version 4 Reference Manual for Disc Systems".

Research Machines Limited is committed to the policy of continuous improvement of its products and services. To assist the Company to improve the quality of its documentation and computer software it would like to receive reports from users on their experience with using them.

Reports referring to documentation should include:

- a description of any errors you have found;
- comments on problems you have encountered with understanding and using the manual, and suggestions for improvement;
- your name and address and a brief description of your experience with using microcomputer systems.

Queries on documentation should be sent to:

The Documentation Department,
Research Machines Limited,
PO Box 75,
Oxford.
OX2 0BW.

Reports referring to software errors should include:

- a written description of each step taken and each key or command you type.

Queries on software errors should be sent to:

Technical Support Department,
Research Machines Limited,
PO Box 75,
Oxford.
OX2 0BW.

TXED VERSION 4

RELEASE NOTE

In order to keep it up to date, the Release Note is supplied as a file RELEASE.PRN on the Distribution disc rather than being bound in with the Manual. The Release Note contains details of the memory addresses used by TXED together with information about any improvements and additions since the Manual was published.

To display the Release Note on the screen, follow the instructions in Chapter 2 for loading TXED but instead of typing 'TXED', type:

B>TYPE RELEASE.PRN

followed by the RETURN key. If you wish to print it (which we recommend) select a printer option first, as described in Appendix F, and type CTRL P (hold down the CTRL key while typing the letter P) prior to typing RETURN.

TEXT FORMATTER

Please note that TXED is sold in two different versions, one including a built in text formatter and the other without it. Unless you have purchased the version that includes the formatter you should disregard the contents of Chapters 7 and 11.

Copyright May 1980

TXED VERSION 4 REFERENCE MANUAL

CONTENTS

Chapter 1	Introduction	1.1
	History of this version	1.1
	How to use this manual	1.1
	Making a working copy	1.2
	Underlining convention	1.2
	Acknowledgements	1.2
Chapter 2	Getting Started with TXED	2.1
	Loading TXED	2.1
	A first look at the keyboard	2.2
	Command and Immediate modes	2.3
	Typing text in Immediate mode	2.5
	Making corrections	2.5
	Long lines (Part 1)	2.5
	Editing	2.5
	Case shifting	2.7
	Text pointer and text buffer	2.7
	Text window	2.8
	Summary	2.9
Chapter 3	Getting Started with Files	3.1
	What are files?	3.1
	File names	3.1
	Writing text to a file	3.2
	Editing a file	3.3
	How it works (Part 1)	3.4
	General advice	3.5
	Summary	3.5
Chapter 4	More on Editing	4.1
	Long lines (Part 2)	4.1
	Changing the text window	4.2
	Pointer movement commands	4.3
	Deletion commands	4.4
	Insertion of lines	4.4
	Finding a character	4.5
	Inserting and finding a marker	4.5
	Moving and copying blocks	4.6
	Summary	4.7

 Copyright May 1980

Chapter 5	More on Files	5.1
	What to do when an edit goes wrong	5.1
	How it works (Part 2)	5.2
	Abandoning an editing session	5.4
	Suppressing the return to the Operating System	5.4
	How to merge two files	5.4
	Editing large files	5.6
	Summary	5.7
Chapter 6	String Search and Replacement	6.1
	Finding a character string	6.1
	String replacement	6.2
	Command repetition	6.3
	Macro commands	6.3
	Summary	6.3
Chapter 7	Introduction to Formatting	7.1
	Getting started	7.1
	Printing the contents of the text buffer	7.4
	Formatted printing	7.4
	Formatting commands	7.5
	General observations	7.9
Chapter 8	Immediate mode Reference Section	8.1
	Immediate mode commands	8.1
	Correspondence between modes	8.4
Chapter 9	Command mode Reference Section Part 1: Editing Commands	9.1
	Command termination and execution	9.1
	Command structure	9.1
	Special forms of the numeric argument	9.2
	Compound commands	9.3
	Iteration	9.3
	Special characters	9.4
	Commands that move the text pointer	9.4
	Commands that cause deletion of text	9.6
	Text modification command	9.7
	Text insertion	9.7
	Search and replacement of text	9.8
	Commands involving text markers	9.9
	Output of text to screen or printer	9.10
	Saving and copying small portions of text	9.12
	Saving and executing a macro command	9.13
	Utility commands	9.14

 Copyright May 1980

Chapter 10	Command mode Reference Section Part 2: File Related Commands	10.1
	File specifications	10.1
	Commands that open and close files	10.2
	Commands that read and write to files	10.6
	Utility commands	10.9
	Common file operations	10.10
	Starting a new file	10.10
	Editing an existing file	10.11
	Editing a long file	10.11
	Making a copy of a file	10.12
	Merging two or more files	10.12
	Splitting a file into two	10.13
	Listing a file on the printer	10.14
	Formatting a long file	10.14
	Saving formatted text in a file	10.15
	Changing a disc during editing	10.15
Chapter 11	Formatting Reference Section	11.1
	General description of formatting	11.1
	Formatting commands	11.2
	When text is filled	11.2
	Breaks	11.3
	Printing a document	11.3
	Commands that define page limits	11.4
	Page numbering (Part 1)	11.7
	Running heading and folio	11.8
	Page numbering (Part 2)	11.10
	Commands that affect the text lay out	11.11
	Starting a new line	11.13
	Starting a new page	11.16
	Indentation	11.17
	Centering and full indentation	11.19
	Miscellaneous commands	11.20
	Leaving space for a diagram	11.22
	Formatting long files	11.23
	Selective output	11.24
	Special features for daisy wheel printers	11.25
Chapter 12	Macro Commands	12.1
	Defining and executing a macro	12.1
	Multiple execution of a macro	12.3
	Comparison of iteration brackets and macros	12.3
	Advanced editing	12.4
	Two column listing	12.4
	Editing a complex macro	12.5
	Standard letters	12.5
Appendix A	Quick Reference Guide	A.1

 Copyright May 1980

Appendix B	Error Messages	B.1
	Operating System error messages	B.1
	Read only error	B.1
	Bad sector error	B.2
	Warning messages	B.3
	Errors that occur during commands	B.4
Appendix C	Glossary	C.1
Appendix D	Making a Working Copy of TXED	D.1
Appendix E	Configuring TXED	E.1
	Configuration by command	E.1
	Saving a configured version	E.2
	Configuration by patching	E.2
	Patching the initial formatter settings	E.3
	Patching the Immediate mode table	E.5
	Default file types	E.7
	Keyboards with ALT MODE	E.8
	Cursor character	E.9
Appendix F	Setting up a Printer	F.1
	General information	F.1
	OKI Microline 80	F.3
	Anadex DP-8000	F.3
	Qume Sprint 5	F.4
Appendix G	Differences from TEX	G.1
Index		I.1

CHAPTER 1

INTRODUCTION

TXED is a text editing program written specially for the 380Z computer. It allows the creation, modification and printing of textual information such as programs, data, documents and letters. All computer systems have a form of text editor, some more primitive than others. Most users don't know what they're missing.

This is the Reference Manual for TXED. It is really two manuals in one, combining a tutorial guide and a reference section containing the more detailed descriptions of the commands. Throughout the Manual the material you will be editing is referred to simply as 'text', by which we mean just a sequence of lines of characters. Text includes computer programs, alphanumeric data, leaflets, and reference manuals. Although designed primarily for editing programs, TXED can also do context searches for text patterns, make global changes, move paragraphs, and split and combine files, features that are useful in the more general field of word processing. A text formatter is included as well, allowing documents to be laid out neatly on a suitable printer.

But perhaps the most striking feature is the 'Immediate' editing mode, in which the keyboard and screen are linked to allow editing by single keystrokes. The beginner can start work at once yet the more experienced user will not become frustrated.

HISTORY OF THIS VERSION

TXED Version 4 was first released in June 1980 and is the most recent version of a program of the same name written for the 380Z in July 1977. Significant changes in the new release include the addition of word and block oriented commands, much improved display and file handling, and the built in text formatter. All the commands of the previous versions have been retained.

HOW TO USE THIS MANUAL

The first part of the Manual is in the form of a Tutorial Guide and it is intended that you work through it at the computer, trying out the examples as you go along.

We recommend that you begin by reading Chapters 2 and 3, after which you will be able to start using the editor. It is best to master the contents of these Chapters before moving on.

Chapters 4 and 5 expand on the material presented in Chapters 2 and 3, introducing additional editing and file handling commands. Chapter 6 considers the problem of searching for a string of characters and of replacing it by another string. Chapter 7 introduces the facilities for formatting text on a printer. Chapter 12 is the last of the tutorial Chapters and covers the important topic of the macro command. It is out

 Copyright May 1980

of order because it requires knowledge of material that is contained only in the Reference Chapters.

The Reference Section of the Manual is contained in Chapters 8 to 11 and is divided into sections dealing with Immediate mode, Command mode, file handling and formatting.

There are also a number of Appendices concerned with individual topics. We draw your attention particularly to Appendix A, the Quick Reference Guide. A second copy of this is provided that you can keep by the computer.

Appendix B contains a list of Error Messages and instructions for recovering from some of the more common error conditions. Look in Appendix B first when something goes wrong.

Appendix C is a Glossary of technical terms used in the Manual.

MAKING A WORKING COPY

One of the first things you should do on receiving TXED from Research Machines is to make a working copy of the Distribution disc or tape and put the original away in a safe place. The procedure for doing this is outlined in Appendix D.

UNDERLINING CONVENTION

As well as being used for Chapter and section headings, underlining is used throughout the Manual for two other purposes:

In the examples in the early Chapters, the characters that you should type are underlined to differentiate them from the ones that the computer will print. This convention is dropped in the Reference Chapters in which it is assumed that you are already familiar with the general forms of the various commands.

Certain important concepts and technical terms are underlined when they are first described, in order to make them stand out more clearly on the page. Similarly, each command is underlined when it is defined, the commands appearing on the left of the page with an accompanying description on the right.

ACKNOWLEDGEMENTS

We would like particularly to acknowledge our debt to Dr Robin Willison and his colleagues at the National Hospitals for Nervous Diseases, Queen Square, London, without whose material help, encouragement and criticism this Manual would not exist.

We are grateful also to our Users who have taken the time to make suggestions for the improvement of previous versions. Many of the good ideas that have been incorporated in the present version of TXED are due to them.

CHAPTER 2

GETTING STARTED WITH TXED

This Chapter introduces some simple ways of using TXED to type text in to the computer and edit it. When you have mastered these you will have the means to tackle almost any editing problem. There are, of course, many more commands in TXED than this Chapter covers, but, like any complex subject, TXED is best learned in stages.

The first stage is to work through the editing commands described in this Chapter at the computer. However, you will not be able to start using TXED seriously until you have also learned how to save and recall text on disc. This topic is covered in Chapter 3.

LOADING TXED

Load TXED into the computer. Switch on the 380Z or press the RESET button. Insert a CP/M System disc in drive A and a TXED disc in drive B, making sure that the discs are the right way up and the drive doors are closed. Type B to boot the system. The computer will print:

```
RESEARCH MACHINES
31K CP/M VERS 1.4

A>
```

or something very similar. Log in to drive B by typing

```
A>B:
```

followed by the RETURN key. You should only type the underlined characters; the computer will print the rest. The computer responds by printing

```
B>
```

Then start TXED by typing its name, again followed by RETURN:

```
B>TXED
```

When TXED has been loaded into the computer, it prints something like:

```
TXED V 4.2 E
*
```

The above would show that you were using TXED Version 4.2 Revision E; it is important to quote the version number if you ever write to Research Machines with a query. At the same time, the upper 20 lines of the screen are cleared.

The asterisk indicates that TXED is ready for you to type a command on the keyboard.

 Copyright May 1980

A FIRST LOOK AT THE KEYBOARD

A number of versions of the 380Z keyboard have been produced. Most of these include the keys that will now be described. If you cannot find a particular key on your keyboard, refer to Appendix E.

Have a look at the keyboard. It is fairly similar to a typewriter one, but there are certain keys that have special significance when editing:

RETURN The RETURN key tells the computer to begin a new line.

SHIFT This key has its normal action. When held down, the letter keys produce capital letters and the number ones the punctuation characters marked above them. There may be an exception in the case of SHIFT 3; this produces the sharp or hash character (#) despite the fact that it is often marked with a pound sign.

CAPS LOCK This key has the action of setting or clearing an electronic lock on the letter keys. Pressing it once locks them so that they always generate capitals, irrespective of whether the SHIFT key is held down or not. Pressing it again clears the lock and allows the letter keys to generate lower case (small letters) when SHIFT is not pressed. Unfortunately, on many keyboards there is no way of telling whether the lock is set or not without typing something.

CTRL Like SHIFT, holding down this key while typing one of the letter keys alters the letter key's action. The key marked "D", for example, has three distinct actions, depending on whether it is typed alone, while SHIFT is held down, or while CTRL is held down. (There are of course only two actions available when CAPS LOCK is active). Keys typed while holding down CTRL are referred to colloquially as the control keys and are used as editing commands.

DELT The DELT key, which stands for delete, rubs out the most recently typed character, having an action fairly like backspace on a typewriter. Some versions of the 380Z keyboard require that the SHIFT key be held down while typing DELT. CTRL DELT has the same effect as DELT and does not require SHIFT to held down.

REPT The REPT key may repeat the last character typed at a rate of about 10 times per second. If it does this on your keyboard, you will find it a useful addition to some of the editing commands to be described later in this Chapter.

ESC This is a particularly important key in TXED, being used to tell the computer that you have finished typing a command and that the computer should now execute it. TXED prints a dollar sign ($) when ESC is typed, but to the computer this is quite distinct from the dollar sign key (SHIFT 4).

 Copyright May 1980

One difference between TXED and almost all other programs will now be apparent. In most programs RETURN is used to tell the computer that you have finished typing some input. In TXED, RETURN is treated like any other character and ESC is used to indicate the end of a command. This may take some time to get used to.

Most of the other keys produce the characters marked on them.

COMMAND AND IMMEDIATE MODES

TXED has two quite distinct modes of operation, referred to as Command mode and Immediate mode. Command mode allows you to enter the more complex editing commands, such as those to save edited text on disc. Immediate mode allows you to type in text to the computer and has provision for simple editing by means of the control keys. After describing how to move from Command mode to Immediate mode, the rest of this Chapter concentrates on Immediate mode. We will return to Command mode in the next Chapter.

TXED is in Command mode when it is first loaded into the computer, and indicates this by printing an asterisk (*). Whenever the asterisk is displayed you can move into Immediate mode by typing the ESC key twice. The two ESCs are printed as dollar signs and TXED responds by printing an exclamation mark (!), indicating that it is now in Immediate mode:

```
*$$
!
```

(Remember that in the examples, you type only the underlined characters).

You can move back from Immediate mode to Command mode at any time by typing ESC once. TXED responds with an asterisk, indicating that it is again in Command mode. Continuing the previous example, typing ESC for a third time would produce

```
*$$
!
*
```

Note that on this occasion the ESC key does not echo as a dollar.

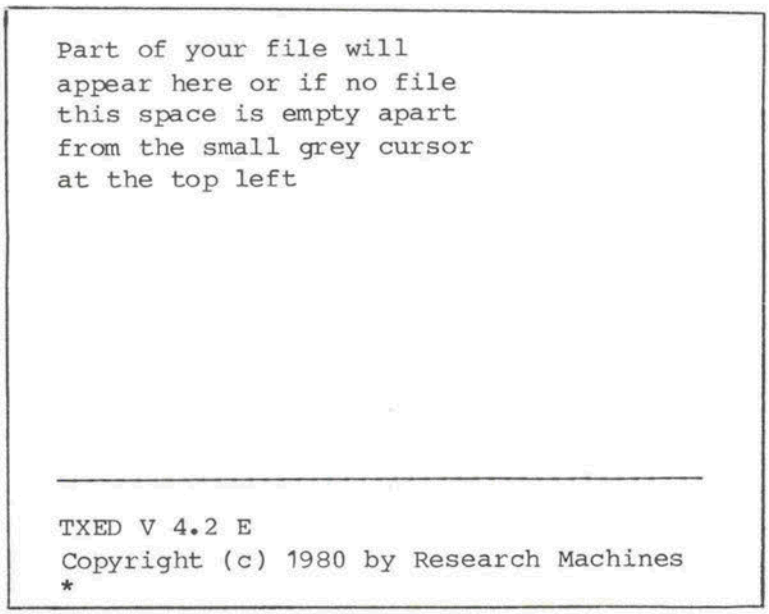
You can also move from Command to Immediate mode by typing an Immediate mode command (see below). This only works if the Immediate mode command is the first character to be typed after the asterisk is printed.

 Copyright May 1980

To start the Editor:

TXED	Starts the editor
TXED filename	Edit the named file

The screen is divided into two at this point as shown below.


Part of your file will
appear here or if no file
this space is empty apart
from the small grey cursor
at the top left
TXED V 4.2 E
Copyright (c) 1980 by Research Machines
*

Symbols

*	Command mode
!	Immediate mode
$	ESC

To transfer between modes, press the ESC key once or twice as appropriate. You will observe a change in the lower section of the screen from ! to * or vice versa.

Fig 2.1 Showing the initial state of the editor. Editing a file is described in Chapter 3.

TYPING TEXT IN IMMEDIATE MODE

Practise moving between Command mode and Immediate mode a few times, then enter Immediate mode (exclamation mark prompt) and type a sentence such as:

```
Pack my box with five dozen(RETURN)
liquor jugs
```

where (RETURN) stands for the RETURN key and begins a new line. Notice that each letter appears on the screen as you type it (hence the name Immediate) and that as you type, the flashing square moves across screen. This square indicates where the next character will appear and is called the cursor.

As suggested, type the RETURN key to move down to the beginning of the next line as you get towards the right hand side of the screen. Notice that RETURN has a similar action to the carriage return key on an electric typewriter. You can type RETURN several times in succession to leave some blank lines.

MAKING CORRECTIONS

The beauty of typing on a computer is that you can always correct your mistakes! When you make an error, all you need do is to type DELT enough times to rub out the offending characters, then replace them with the correct ones.

If you haven't made a mistake yet, make some deliberate ones and practise correcting them.

LONG LINES (PART 1)

If you are a good typist you may have already "gone off" the right hand side of the screen. If not, try typing a long line now. You will find that when you get near to the right hand edge, the line of text moves bodily to the left, to keep the cursor on the screen, and that the beginning of the line disappears. TXED puts a special arrow character at the start of the visible line to show that this has happened. But when you type RETURN, the beginning comes back into view, and the end of the line disappears to the right (it is still there). Rub out the RETURN with DELT and you are back at the end of the line.

We will return to the problem of long lines in Chapter 4. For the moment, however, you will find TXED easier to understand if you confine yourself to lines no longer than the screen width.

EDITING

So far, we have seen how to correct mistakes by rubbing them out with DELT. This is effective enough but could be annoying if you have typed several more words before spotting the error. The advantage of using a computer for editing is that it allows you to move about in the text to

make corrections and alterations without affecting the intervening characters.

We will now describe four such commands which will allow you to accomplish almost any editing task. When you begin to find them tedious it is time to move on to Chapter 4.

The four commands are all control characters:

CTRL D Move forward by one character.

CTRL V Move backward by one character.

CTRL N Move forward by one line.

CTRL G Move backward by one line.

Remember that to type a control character, you first hold down CTRL and then type a letter (while keeping CTRL held down). It doesn't matter whether the letter is in capitals or lower case. Think of CTRL as being similar to SHIFT.

Get used to moving around the screen and making changes in what you have typed. If you move the cursor to between two characters, you can type in additional characters. The existing text moves apart to accommodate them. If you don't like what you have added, you can rub it out again with DELT and the text will close up. Notice that DELT always rubs out to the left, so you will first have to position the cursor to the right of any characters that you want to erase.

Observe that CTRL N always moves to the beginning of a line but that you can't move further down than the end of the text, and that CTRL G moves to the beginning of a line if you are in the middle of it, or up a line if you are already at its start. CTRL G won't take you higher up the page than the beginning.

Using CTRL D and CTRL V and perhaps DELT, confirm that the RETURN character that you typed at the end of each line is treated like any other character. Thus CTRL D moves from the end of one line to the beginning of the next and CTRL V moves back again, and if you are at the beginning of a line and type DELT, the RETURN at the end of the previous line is rubbed out and the two lines become one.

By moving the cursor to the middle of a line and typing RETURN you can begin a new line wherever you want. Try this out.

 Copyright May 1980

CASE SHIFTING

So far, we have been able to insert and delete text, and to move the cursor around. We now introduce a command that modifies the text.

CTRL S Change from capitals to lower case, or vice versa.

With the cursor at the beginning of line, type CTRL S a few times to see its effect. Move the cursor back to the beginning of the line with CTRL G and type CTRL S again to reverse the changes you have made.

Although the CTRL S command is not used particularly often, we have mentioned it now because, in contrast to DELT which deletes the character before the cursor, CTRL S changes the next character. As you gain experience with TXED you will need to make the distinction between commands like DELT which act backwards, others like CTRL S which act forwards, and even some which can act in either direction.

TEXT POINTER AND TEXT BUFFER

Everything you have typed so far has had an immediate effect and we hope has seemed fairly natural. However if you pause to think about it, you will realise that everything that has happened on the screen has been in relation to the flashing cursor. Typing a letter inserts it at the cursor position. Typing DELT deletes the character immediately before the cursor. Typing CTRL S swaps the case of the character after the cursor. The four movement commands move the cursor around.

In formal terms, we say that the cursor represents the position of the text pointer within the text buffer.

A buffer is just a name for an area inside the computer which can store things. The text buffer is where your text is stored. The computer keeps track of where you are working within the text buffer, storing this information as the text pointer. Clearly, before you have typed anything, the text buffer is empty and the text pointer points to its beginning.

It is important to realise that the text pointer points between two characters. This is difficult to show clearly on the 380Z screen but if you move the pointer to the middle of a word and examine the flashing cursor carefully, you will see that the grey rectangle lies nearer to the beginning of the letter it lies over than to its end. The text pointer position is in fact between that character and the one immediately before it.

When the pointer is at the end of a line (strictly speaking, between the last character of the line and the RETURN character which terminates it), the cursor changes from a rectangle to a square.

 Copyright May 1980

TEXT WINDOW

What you see on the screen is a picture of part of the text buffer, with the position of the text pointer shown by the cursor. This is called the text window. The text buffer can hold a lot more text than can be shown on the screen. TXED selects what to display by trying to keep the text pointer about half way down the screen, except when the pointer is near the beginning of the text buffer.

If you have not yet typed in more than about eight lines of text, you have probably had the impression of the cursor moving up and down the screen as you type the CTRL N and CTRL G commands. Once you have typed in more than 20 lines or so, you will see that a more accurate impression is of the text moving up or down past the pointer, hence the term window.

TXED has several commands available to alter the behaviour of the text window (see Chapters 4 and 9).

 Copyright May 1980

SUMMARY

The editing commands described in this Chapter are:

ESC ESC Enter Immediate mode from Command mode.

ESC Return to Command mode from Immediate mode.

CTRL D Move text pointer forward one character.

CTRL V Move text pointer backward one character.

CTRL N Move text pointer forward one line.

CTRL G Move text pointer backward one line.

CTRL S Swap case of character after the text pointer.

DELT Delete character before the text pointer.

In Immediate mode, most characters typed are inserted at the position of the text pointer.

As mentioned earlier, this small group of commands is sufficient to tackle almost any editing problem. For this to be anything other than tedious, however, it is important to learn the four cursor movement commands thoroughly. Many users have found a little diagram such as the following useful:

CTRL-	Chars	Lines
Forward	D	N
Back	V	G

Later you will find the Quick Reference Guide in Appendix A is all you need to refresh your memory, but for the moment it probably contains too much information.

If you find our choice of control characters impossibly arbitrary, there is the possibility of tailoring them to your personal preferences (see Appendix E). We suggest you defer making a final decision on this until you have reviewed all the available commands.

You will have to study the next Chapter before starting work with TXED but when you have done so you will have covered all you need to know to get started.

 Copyright May 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

CHAPTER 3

GETTING STARTED WITH FILES

The purpose of this Chapter is to introduce some of the commands for storing text on floppy discs, and for recalling it back into the computer memory.

When you have typed a piece of text into the computer, and don't want to lose it when you turn the computer off, you must store it in a file on a disc. The information you have stored stays in that file until you erase it, and can be recalled back into the computer at any time for further editing.

WHAT ARE FILES?

A file is simply an area of storage on a disc, usually containing a single program, an item of text, or a set of data.

TXED has a number of commands which cause text to be stored in a file. Such text must have previously been typed in to the text buffer, or it must be present in another file from which it can be transferred. The process of storing information in files is called writing to files.

TXED also has commands to perform the reverse operation, reading from files. This process retrieves text from a file and makes it available for further editing.

FILE NAMES

When you store some information in a file you must give the file a name, so that you can refer to it again.

A file name consists of three parts, a disc name, the file name itself, and the file type, making up a file specification. A typical file specification might be:

The disc name is a single letter, followed by a colon (:), indicating the disc unit on which the file is to be stored. This is generally in the range A to D, but depends on the disc system you are using. Consult your System Manual if you do not already know how many disc units you have.

The file name consists of 1 to 8 letters or numbers. It is usually chosen to indicate the file's purpose:

```
LETTER1
CHAPTER2
STATS
```

The file type consists of up to 3 letters or numbers, separated from the file name by a full stop (.). Certain file types are standard on the 380Z, for example:

ALG	Algol program
BAK	Backup file
BAS	BASIC program
FOR	Fortran program
PRN	Program listing
TEX	Text file
ZSM	Assembly Language program

but these are usually only for convenience, and you can invent your own file types if you wish, or omit them entirely.

Further examples of file specifications are:

```
A:LETTER1.TEX
B:STATS.BAS
CHAPTER3
```

In the last example both the disc name and file type have been left out.

If no disc name is given, the system assumes that the file is on the 'logged in' disc, i.e. the disc whose name appeared in the CP/M prompt when TXED was loaded. If you have been following the examples exactly, this is currently disc B (see Chapter 2).

A file specification must be unique. No two files on the same disc may have the same name and type. It is possible to have a group of files on a disc with the same name and differing types. Thus you might have both LETTER1.TEX and LETTER1.BAK on disc B. The implications of this will be explained below.

WRITING TEXT TO A FILE

We will begin by describing the simplest way of saving your text in a file.

If you are still in TXED, press the RESET button and start again.

Follow the instructions for loading TXED in Chapter 2 as far as the point where the computer prompts B>, then load TXED by the command:

```
B>TXED B:TEST1.TEX
```

followed by RETURN. TXED is loaded into memory, and announces itself as before:

 Copyright May 1980

```
TXED V 4.2 E
New file
*
```

but this time assumes that you wish to edit the file B:TEST1.TEX. Since no file of that name yet exists, TXED creates it and informs you that it is a new file.

Enter Immediate mode (ESC ESC) and type in one or two sentences. Then return to Command mode (ESC).

You now have some text in the text buffer. To save it in a file, we need a new command, one that can only be given in Command mode:

EX End an editing session, returning to the Operating System.

To issue this command, you need merely type the letters EX and then the ESC key twice:

```
*EX$$
```

where $ stands for ESC.

The EX command causes TXED to write the contents of the text buffer to a disc file (B:TEST1.TEX in this case) and then exit to the Operating System.

Notice that EX is followed by two ESCs rather than by RETURN. In Command mode the first ESC terminates a command and the second causes it to be executed.

You should now be back in the Operating System (B> prompt). To confirm that you really have created a file, give the Operating System command DIR:

```
B>DIR
```

(followed by RETURN). This command lists the names of all the files on disc B on the screen. TEST1.TEX should be among them.

EDITING A FILE

Again, the simplest way to edit a file is just to type:

```
B>TXED B:TEST1.TEX
```

(then RETURN). TXED is loaded as before but this time the 'New file' message doesn't appear. Instead the contents of TEST1.TEX are read into the text buffer and displayed on the screen, ready for editing.

Confirm that this is what you typed in earlier, then enter Immediate mode and make a few alterations. Return to Command mode and again write out the text to disc with the EX command (EX ESC ESC).

 Copyright May 1980

We suggest you repeat this procedure a few times until it becomes familiar and until you are convinced that all your changes have been included each time. If you like, try turning the computer off between edits (take the discs out first) to confirm that you really are reading and writing text to disc.

By gaining confidence in what should happen, you will more easily recognise when something goes wrong and will probably be able to work out how to recover from your error.

HOW IT WORKS (PART 1)

It should be fairly clear by now that editing is carried out in the text buffer in the computer's memory. When a file is first created, text typed at the keyboard is stored in the text buffer and can be edited or saved in a file, as shown in Figure 3.1:

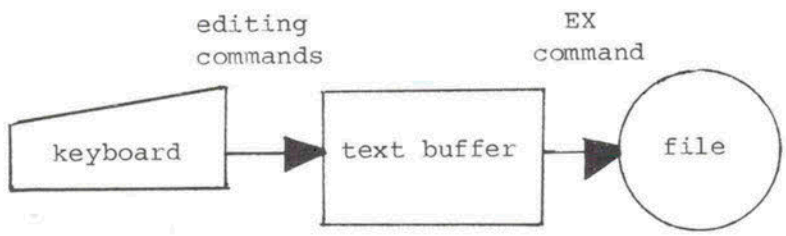


Fig 3.1 Simplest form of text entry, editing and creation of an output file (identified by a file specification given when TXED was loaded).

When a file is edited, the process can be represented as in Figure 3.2:

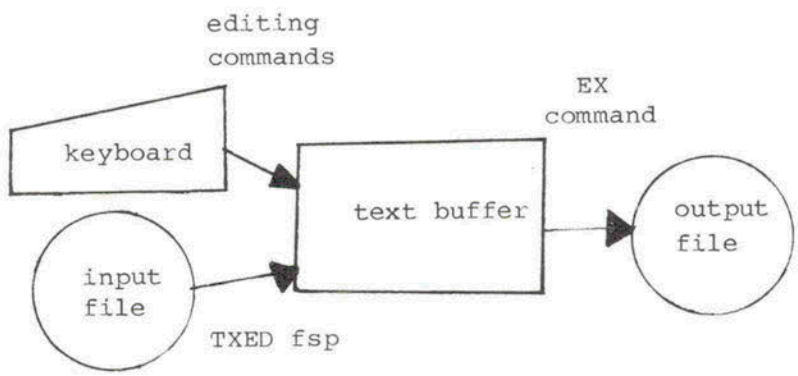


Fig 3.2 An input file is read into the text buffer, it is edited by means of commands and the addition of text, both entered on the keyboard. As in Fig 3.1 the final result is an output file. Both files are identified by file specifications which must differ if they are contained on the same device. This is handled automatically by the text editor.

 Copyright May 1980

At the end of an editing session (i.e. after the EX command) there will in fact be two files on disc B, TEST1.TEX and TEST1.BAK. You can confirm this with the DIR command. TEST1.BAK contains the state of TEST1 before the most recent edit and TEST1.TEX its state afterwards.

This process, known as 'backup', allows you to return to the previous version of the text if an editing session goes badly wrong. We will return to this topic in Chapter 5.

GENERAL ADVICE

Never type away for hours without saving your text on disc.

If something can go wrong, it will. Causes of disaster range from simply being distracted by a telephone call to losing power to the computer due to someone tripping over the mains lead or to a power cut.

It takes much less time to save your work on disc every so often and then re-edit it, than to type in a long edit again.

Important files should always be copied onto a second disc and stored safely.

If you are writing a long text in several sessions you can save the work of each session in a separate file, each with a serial number, for example, BOOK1, BOOK2, etc. Such smaller files are more manageable and they can always be combined into a single file later if necessary.

SUMMARY

Only two new commands have been presented in this Chapter:

TXED fsp Edit (or create) the file named by the file specification fsp.

EX End an editing session, returning to the Operating System.

In addition, we mentioned the Operating System command

DIR List the file directory on the screen.

These commands and the ones in the previous Chapter will allow you to start work.

If you are planning to use TXED to type in and edit programs, we suggest that you start by writing a small program now and then try running it. Many users find the facilities described so far adequate for almost all their work.

If you intend using TXED mainly for text processing, you should now type in one or two short letters and practise editing them. It is important to become thoroughly familiar with the contents of the last two Chapters before proceeding, but for a taste of things to come, you might like to glance at the Introduction to Text Formatting in Chapter 7.

 Copyright May 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

CHAPTER 4

MORE ON EDITING

In Chapter 2 we described Immediate mode commands to move the pointer and swap case. These five commands are sufficient for occasional editing but become tedious when much work needs to be done. We now introduce a number of additional commands. Each is useful but you will probably not start using all of them regularly until you know TXED quite well. Before doing so, however, we will continue the discussion of how to deal with long lines.

LONG LINES (PART 2)

In Chapter 2 we noted that when a long line was typed and as the cursor approached the right hand side of the screen, the line containing the cursor was shifted to the left to keep the cursor on the screen. We advised you to avoid long lines at that time, the reason being that it is much easier to edit a piece of text when you can see all of it.

This initial mode of the text window, in which long lines are allowed, is in fact intended mainly for entering line oriented text such as computer programs.

When you wish to type in a document (for example, this Manual) the mode has the disadvantage that you have to keep an eye on the screen all the time if you want to avoid typing a long line. Unlike a typewriter, there is no bell to warn you when you are getting near the edge of the screen.

The QS command allows you to set a right 'margin' which applies only while typing in text:

nQS Set automatic right margin

This command sets an automatic right margin that applies when typing in text in Immediate mode. The margin can be set to any column in the range 20 to 200 (20QS to 200QS). When the cursor passes this column, the next space character to be typed is converted to a RETURN character, forcing text entry onto a new line but only breaking a line on a word boundary.

The QS command is given in Command mode (after an asterisk prompt). For example, for a 40 character screen, an automatic margin at column 32 will be found convenient. To set this, type:

*32QS$$

where $ stands for the ESC key. 70QS should be used with an 80 character screen. Now, when you enter Immediate mode and start typing in text, you will find that as the cursor approaches the right hand side of the screen the computer begins a new line automatically at the end of a word.

 Copyright May 1980

To return to the initial setting, enter Command mode and type:

*200QS$$

Even when typing a document, you will need to return to the initial state (where the right margin is set to column 200 and effectively inoperative) from time to time, for example, when typing in a table.

CHANGING THE TEXT WINDOW

Sometimes it is necessary to be able to edit long lines, for example, when editing a Cobol program. TXED has two other forms of display that are helpful, both controlled by the EL command:

nEL Modify text window display

This command allows the behaviour of the text window to be changed.

When n is zero (0EL), the initial form of display is selected. Only the first 40 characters of a line (or 80 on an 80 character screen) are displayed, except in the case of the line containing the cursor. The cursor is always displayed, the line containing it being shifted to the left if necessary.

When n is positive (+1EL), the whole of each line is displayed, long lines being continued on subsequent lines. A special graphics symbol is displayed at the beginning of each continuation line. (Note that in this mode, TXED still considers a 'line' to be a series of characters terminated by the RETURN character even though it occupies several lines on the screen. Thus the CTRL N command, in moving the text pointer forward by one logical line, may move it down several screen lines.)

When n is negative (-1EL), 'horizontal scrolling' is enabled. This is similar to the initial mode in that only 40 (or 80) characters of each line are displayed, but as the cursor nears the right hand side of the screen the whole display slides to the left allowing changes being made to the end of a long line to be seen in the context of the preceding and succeeding long lines (if any). This mode is particularly suitable for editing tables.

The EL command is given in Command mode. Examples are:

*+1EL$$	Continue long lines
*-1EL$$	Select horizontal scrolling
*0EL$$	Return to initial mode

We recommend that you spend some time trying out these various forms of display to discover your own preferences. Most people use the combination of 0EL with 200QS for editing programs and 0EL with 32QS for

documents, reserving the other forms of the EL command for special occasions.

Further details of the QS and EL commands will be found in Chapter 9 in the Utility Commands section.

POINTER MOVEMENT COMMANDS

Returning to the commands used in editing, there are 10 Immediate mode commands to move the pointer, including the four already described in Chapter 2:

CTRL D Forwards one character.

CTRL V Backwards one character.

CTRL W Forwards one word.

CTRL U Backwards one word.

CTRL N Forwards one line.

CTRL G Backwards one line.

CTRL P Forwards one page (15 lines).

CTRL Q Backwards one page (15 lines).

CTRL Z Move to end of text.

CTRL B Move to beginning of text.

Movement by characters and by lines has already been described in Chapter 2.

The word movement commands are set up initially to move the pointer to the end of the next or preceding word. In effect a word is treated as though it is a single character. These two commands are useful for stepping quickly along a line. An alternate definition of a word, which leaves the pointer at the beginning of a word, can be selected by the QU command (see Chapter 9). We suggest that you try out both definitions and use the one you prefer.

CTRL W and CTRL U behave rather peculiarly near the end and beginning of

the text buffer. This is not a program 'bug', merely a consequence of the way in which a word is defined.

The CTRL P and CTRL Q commands, which move the pointer by 15 lines in either direction, are useful for stepping quickly through text to get to a given place, or for reviewing changes in a wider context. Note that the distance moved slightly overlaps; you will not miss a line by stepping through with CTRL P (except when the +1EL mode of display has been selected).

The effects of CTRL Z and CTRL B are self evident.

DELETION COMMANDS

In addition to the DELT key, there are three other commands that delete text:

CTRL E Delete character forwards.

CTRL R Delete word backwards.

CTRL K Delete line forwards.

The action of CTRL E is similar to that of the DELT key except that the character after the text pointer is deleted rather than the one just before it. DELT is useful when typing in text as you usually want to delete something you have just typed, but when editing you may find it more natural to be able to delete forwards.

CTRL R deletes both one word and one lot of punctuation characters between words. Exactly what happens depends on the word definition currently in force. If the word definition is such that the pointer comes to rest at the end of a word, then CTRL R will delete back to the end of the preceding word, including any punctuation characters encountered in getting there. If the alternate word definition has been selected, the command deletes back to the beginning of a word, deleting at most one lot of punctuation characters.

If the pointer is at the beginning of a line, CTRL K deletes the whole of that line, including the RETURN character that terminates it. If the cursor is in the middle of a line, the rest of the line to the right of the pointer, again including RETURN, is deleted.

INSERTION OF LINES

You have probably noticed that you can insert additional lines of text in the middle of a section by moving the pointer to the beginning of a line and typing RETURN a few times to open up a gap. The only problem with this is that it leaves the pointer at the end of the gap, rather than at its natural place at the start. The command

CTRL J Insert blank line.

is provided to cater for this. The CTRL J command is particularly useful when text is being edited on a line by line basis, such as in writing a program.

Since the code generated by typing CTRL J is LINE FEED, you can use the LINE FEED key instead if you prefer.

FINDING A CHARACTER

Sometimes none of the pointer movement commands is particularly convenient for getting where you want to go. The CTRL Y command will move the pointer to just after any character you specify. Its definition is

CTRL Yc Move the pointer forwards to just after character c, where c is any character.

Thus, to move the pointer to the end of a sentence, you would type CTRL Y and then a full stop.

CTRL Y is a particular instance of a much more general character search mechanism which is described in Chapter 6.

INSERTING AND FINDING A MARKER

The problem with CTRL Y is that many characters occur fairly frequently. It would be tedious to have to move to the end of the fifteenth sentence by typing CTRL Y full stop fifteen times. One can always reserve a rarely used character such as ampersand (&) as a marker, but in fact TXED has a specific text marker character and mechanisms for inserting, finding, and deleting it.

A marker is inserted at the text pointer position by typing CTRL O:

CTRL O Insert text marker.

The marker is displayed as the vertical bar character (||) but to the computer it is quite distinct from this.

A special form of the CTRL Y command, CTRL Y followed by the ESC key, finds the next text marker, working forwards from the current text pointer position.

You can thus mark a position in the text that requires further attention, move elsewhere to carry out other editing, then return to the marked point in a straightforward fashion. You may need to move the pointer to the beginning of the text buffer first if you are working past the marked

position.

You are not limited to one marker and typing CTRL Y ESC repeatedly will find a series of markers in succession.

The text marker character can be deleted like any other character (by DELT etc.).

Note, however, that all text markers are removed when text is written to a file. They will no longer be present when the file is read back into the text buffer.

MOVING AND COPYING BLOCKS

It is sometimes desirable to move a block of text, e.g. a paragraph, from one point in a document to another.

If a pair of text markers is placed around a block of text, that block can then be moved to another position, copied to another position, or deleted.

In the move and copy operations, the destination is the position of the text pointer.

The commands to move, copy, or delete a block of text are given in Command mode. The commands are:

QM Move marked text.

This command moves the block of text between two markers to a position immediately before the text pointer. The pointer must be before or after both markers, not between them, otherwise the error message

?Markers?

will be displayed and no move will take place. The markers are deleted after a successful move.

QC Copy marked text.

This command copies a block of text delimited by two markers, inserting the copy immediately before the text pointer. As in the QM command, the pointer must be outside the marked block. Because this command generates extra text there must be enough room in the text buffer to hold it, otherwise the error message

?No room?

will be displayed. The markers are not deleted.

QK Delete marked text.

This command deletes a block of text between two markers and deletes the markers.

The length of a marked block can vary from one character to the whole of the contents of the text buffer.

More than one block of text can be marked. All three commands operate on the first block found, starting from the beginning of the text buffer. In the case of the QM and QK commands, which also delete the markers, several blocks may then be moved or deleted in succession.

To try out these commands, type in a few lines of text in Immediate mode, then choose one of them to move or copy. Move the text pointer to the beginning of your chosen line and insert a text marker by typing CTRL O. Move to the start of the next line and insert a second marker. Finally move the pointer to where you want to move or copy the block and leave Immediate mode by typing ESC.

You are now in Command mode and can issue QM, QC or QK commands, remembering to terminate each command by typing ESC twice. It is worth experimenting thoroughly with each command to gain a good understanding of what it can do. Some more commands concerned with markers are described in Chapter 9.

SUMMARY

In this Chapter and Chapter 2 we have covered the majority of editing commands that you will use most commonly. The Immediate mode commands are:

CTRL B Move text pointer to beginning of text.

CTRL D Move text pointer forwards one character.

CTRL E Delete character after text pointer.

CTRL G Move text pointer backwards one line.

CTRL J Insert blank line after text pointer.

CTRL K Delete line after text pointer.

CTRL N Move text pointer forwards one line.

CTRL O Insert text marker before the text pointer (CTRL Y ESC finds next text marker).

CTRL P Move text pointer forwards one page.

CTRL Q Move text pointer backwards one page.

CTRL R Delete word before text pointer.

CTRL S Swap case of character after text pointer.

CTRL U Move text pointer backwards one word.

CTRL V Move text pointer backwards one character.

CTRL W Move text pointer forwards one word.

CTRL Yc Move text pointer forwards to just after character c, where c is any character.

CTRL Z Move text pointer to end of text.

In addition, the following marker related commands have been described. These must be issued in Command mode:

QM Move marked text.

QC Copy marked text.

QK Delete marked text.

The two remaining commands are also given in Command mode and alter the behaviour of the text window:

nQS Set automatic right margin to column n, where n can lie in the range 20 to 200.

nEL Modify text window display to mode n. If n is zero, truncate lines at right hand edge of screen. If n is positive, continue long lines. If n is negative, enable horizontal scrolling.

You may now like to make yourself another diagram that includes the new commands, along the lines of the one suggested at the end of Chapter 2:

Moving the Pointer:

CTRL-	Chars	Words	Lines	Pages	Buffer
Forward	D	W	N	P	Z
Back	V	U	G	Q	B

Deletion:

CTRL-	Chars	Words	Lines
Forward	E		K
Back	DELT	R	

You should also have a closer look at the Quick Reference Guide in Appendix A. You will find that most of the commands on the first page have now been covered.

CHAPTER 5

MORE ON FILES

In this Chapter we introduce some further commands for handling files. They are all issued in Command mode.

We will take the approach of posing some typical file handling problems and illustrate how to deal with them. In this way, you will learn the new commands in the context of what they can do for you, rather than as abstract definitions.

We may not describe everything that a particular command can do in this Chapter; for that you will have to refer to Chapter 10. We will certainly not be describing everything you can do with a series of file handling commands, but we hope that after reading this Chapter you will be able to work out some of these for yourself. Chapter 10 also includes definitions of some of the terms used in file handling, such as 'opening' and 'closing', and a number of recipes for common file handling operations.

The problems we will consider are:

What to do when an edit goes wrong.
How to merge two files into one.
How to deal with large files.

WHAT TO DO WHEN AN EDIT GOES WRONG

Chapter 3 described how to create and edit files. TXED is started by a command of the form:

B>TXED fsp

where fsp is a file specification. Issuing the EX command in Command mode writes the edited file back to disc and returns to the Operating System.

We mentioned briefly that after editing a file several times, there were in fact two files on the disc, called in our example TEST1.TEX and TEST1.BAK. TEST1.TEX includes the results of the most recent editing session, while TEST1.BAK is the state of the file immediately before the last edit.

Suppose you have just carried out a long edit and EXited, and then realise that you didn't really want to make those changes after all. Instead you have some other changes to make.

The command that you would like to give is:

B>TXED TEST1.BAK

but unfortunately this is illegal. It would result in having two files

 Copyright May 1980

of the same name on one disc. Try it if you like; TXED prints the error message:

?Illegal extension?

(extension means the same as file type) and does nothing.

The steps we need to take are:

1. Erase the 'bad' file TEST1.TEX.
2. Rename the 'good' TEST1.BAK to TEST1.TEX.
3. Edit the new TEST1.TEX.

To do this, we need several additional commands:

QL$ List the file directory on the screen (this command also suppresses the text window).

ED Restore the text window.

QEfsp$ Erase the file named by the file specification fsp.

QRnewold Rename the file named by the file specification 'old' to the file specification 'new'.

EBfsp$ Edit the file named by the file specification fsp.

All these commands are issued in Command mode and must therefore be terminated by typing the ESC key twice.

Notice the occurrence of the dollar sign ($) in the definitions of some of the commands, for example, EBfsp$. This stands for the ESC key and illustrates the important point that commands that are followed by an arbitrary string of characters such as a file specification need an ESC at the end to tell the computer where the character string ends.

The sequence of commands to recover from a bad edit is as follows, with two dollar signs indicating that you type ESC twice:

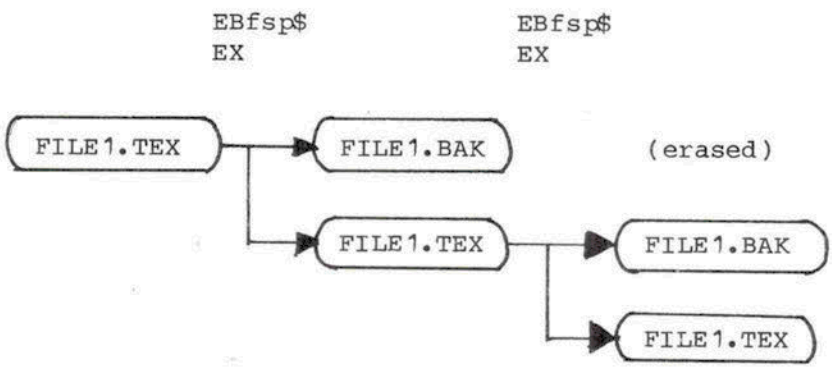
1.	TXED	Load TXED.
2.	QL$$	List the directory of files on the screen to confirm that TEST1.TEX and TEST1.BAK are present.
3.	ED$$	Restore the display window.

4.	QETEST1.TEX$$	Erase TEST1.TEX.
5.	QRTEST1.TEX$TEST1.BAK$$	Rename TEST1.BAK to TEST1.TEX.
6.	EBTEST1.TEX$$	Edit TEST1.TEX.
7.	Add and edit new text	
8.	EX$$	Update and close files, return to Operating System.

The EBfsp$ command is identical to the command TXED fsp, except that it can only be used if the file described by fsp already exists. TXED fsp can also create a new file.

HOW IT WORKS (PART 2)

As mentioned in the previous paragraph, there are two ways of specifying the file you want to edit, by including its name after the Operating System command TXED or in the editing command EB. Both forms of command initiate a process known as 'edit backup'. The advantages of this process are that all the file handling is done automatically and that you always have a second copy of the file in its state prior to the most recent edit. The backup process can be represented as follows:



This diagram makes it clear that at the end of each edit the original file name (FILE1.TEX) is passed on from generation to generation of the file, the file type .BAK being given to the input file for that edit. In human terms, the child carries the family name, the parent is renamed to type .BAK, and the grandparent is erased.

Bearing in mind that all editing is carried out in the text buffer, how is this achieved? When a file is edited, the specified file is opened for reading and its contents copied into the text buffer (the contents of the file itself are not changed). A temporary file is opened for output with the file type .$$$. In the above diagram, this would be called FILE1.$$$ and any existing file of that name would first be erased. Editing proceeds. When the EX command is given, the contents of the text buffer are written to the temporary file and the temporary file is closed. The input file is then renamed to file type .BAK, any other file

of that name (the grandparent) having first been erased. Finally the file type of the original file (which can be any legal type except .BAK or .$$$) is transferred to the temporary file.

Notice that all the renaming takes place at the end of the editing session. This means that if the session is abandoned at any point prior to giving the EX command the files will be as they were before the session started (except that the temporary .$$$ file will have been created). This is an important point; as well as allowing you to have second thoughts, it means that if the power fails during an editing session the damage will usually be confined to losing the current edit. In most cases the files will be left intact.

ABANDONING AN EDITING SESSION

An editing session can of course be aborted by pressing the RESET button or even by turning the computer off. (Remember that you should remove your discs from the disc drives before turning off the power.) The CTRL C command allows you to abandon an editing session rather more elegantly:

CTRL C Return to Operating System

The CTRL C command works in both Immediate and Command modes. Since it is annoying to type it by accident, it asks you:

CTRL C - are you sure (Y/N)?

Answer Y or y if you are; anything else will be ignored. The CTRL C command leaves the editor immediately without closing or renaming files.

SUPPRESSING THE RETURN TO THE OPERATING SYSTEM

Because we can now specify the file we want to edit from within TXED by using the EBfsp$ command, it would be convenient not to have to return to the Operating System and restart TXED every time we type EX. The QX command allows this:

QX Suppress return to Operating System after EX.

The QX command can be issued (in Command mode) at any time prior to using the EX command, and need only be issued once. It is particularly convenient when you need to edit a number of small files. Remember to terminate it with two ESCs.

HOW TO MERGE TWO FILES

Having created a number of short files, perhaps containing frequently used program subroutines or parts of a Chapter, it is often necessary to

merge them into one larger file. We need to be able to specify the files that will be read into the text buffer separately from the file to which the text buffer contents will be written. Some further file handling commands are required:

ERfsp$ Open the file specified by fsp for reading.

R Read from the currently open input file.

EWfsp$ Open the file specified by fsp for writing.

Note the distinction between ERfsp$, which only makes a file available for reading, and R, which carries out the actual work of reading text from it into the text buffer.

Similarly, EWfsp$ only makes a file available for writing. We will use the EX command to do the actual work of writing out the text.

Assuming we have two files, FILE1.TEX and FILE2.TEX, that we wish to combine into a new file FILE3.TEX, the following sequence of commands can be given. All except the first are given in Command mode:

1.	TXED	Load TXED.
2.	ERFILE1.TEX$$	Open FILE1.TEX for reading.
3.	R$$	Read FILE1 into the text buffer.
4.	ERFILE2.TEX$$	Open FILE2.TEX for reading.
5.	R$$	Read in FILE2. This is added on to the end of FILE1.
6.	EWFILE3.TEX$$	Open FILE3.TEX for writing.
7.	QX$$	Suppress return to Operating System after EX (optional).
8.	EX$$	Write text buffer contents to FILE3, then return to Operating System or restart TXED.

It is important to realise that text read in by the R command is always added on to the end of any text that is already stored in the text buffer, irrespective of the position of the text pointer.

If, while editing a document, you wish to merge in a paragraph taken from another file, you will first have to add it on to the end by the ERfsp$ R sequence, then mark it with text markers and finally move it to its destination with a block move (see Chapter 4).

EDITING LARGE FILES

The procedures described in the previous section work without difficulty as long as all the files to be merged are small enough to fit into the text buffer together. The situation becomes more complicated when they grow too large to fit.

In the simplest case, you are editing a file in the normal way, having started off by typing:

```
B>TXED FILE4.TEX
```

to edit the file FILE4, when, having typed in some additional text, the message:

```
?No room?
```

is displayed at the bottom of the screen. The message is telling you that the text buffer is very nearly full. You still have some more text to add. What should you do?

The safest thing to do is to write out the text buffer to file with the EX command. The file FILE4.TEX is now about the same size as the text buffer (the text buffer has a smaller size than the potential capacity of a file, even when the file is on a mini disc).

When TXED reads a file into the text buffer, it aims to fill it only 3/4 full to allow some room for additions. Thus when you next edit FILE4.TEX, TXED will not read the whole of it in and will indicate this by the message:

```
More on input
```

At this point, although you only have access to part of the file, you can continue your editing of the first part. The second part is still on disc waiting to be read. When editing of the first part is complete, you issue the N command to read in the next part of the file:

N — Write out contents of text buffer to output file, clear text buffer, then read in next part of input file (if any).

Thus to get access to the second part of the input file, you need merely return to Command mode, type N ESC ESC, wait for the file operations to finish, then carry on editing.

If you do not wish to make any changes to the second part of the input file, you can issue the EX command instead of the N command. As well as writing out the text buffer contents, EX copies any remaining input to the output file before closing it.

The N command thus allows you to deal with files as large as you like, far larger than would fit into the text buffer.

The disadvantage of this is that you only have access to part of the file

 Copyright May 1980

at a time; it is slightly annoying to be finishing work on the end of a large file, only to remember a change you had to make at the beginning! For this reason, it is advantageous to divide your work into sections small enough to fit into the text buffer in their entirety if you can.

You can run out of space in exactly the same way while using the ERfsp$ R sequence to merge files, and the way round it is the same. Just issue the N command to write out the text buffer, and read in any outstanding input, then continue input.

SUMMARY

The following commands have been presented:

QL$ List the file directory on the screen (turns off the display window).

ED Restore the display window.

QEfsp$ Erase the file named by the file specification fsp.

QRnewold Rename the file named by the file specification 'old' to the file specification 'new'.

EBfsp$ Edit the file named by the file specification fsp.

CTRL C Return immediately to Operating System.

QX Suppress return to Operating System after EX.

ERfsp$ Open the file specified by fsp for reading.

R Read from the currently open input file.

EWfsp$ Open the file specified by fsp for writing.

N Write out contents of text buffer to output file, clear text buffer, then read in next part of input file (if any).

In addition, we used one command from Chapter 3:

EX Write the contents of the text buffer and any remaining text in the input file to the output file. Close the output file. Return to the Operating System unless the QX command has been issued.

These commands enable you to edit, erase, rename, and merge files, and to deal with files larger than the text buffer. They account for the majority of file operations that you will want to carry out on a day to day basis.

The Reference section on files in Chapter 10 includes additional details about the commands described in this Chapter, as well as a few other file handling commands that are needed less often.

 Copyright May 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

CHAPTER 6

STRING SEARCH AND REPLACEMENT

One of the more powerful features of a computer based editor is its ability to search through a document for a given string (sequence) of characters.

Imagine that you had written a long report about Mr Smith. Then you find out he actually spells his name Smythe. It would be very convenient if you could just tell the computer to 'change all the Smiths to Smythe'. TXED has commands to do this.

FINDING A CHARACTER STRING

A string is just a sequence of characters. The process of looking for a particular string of characters is known as string search. A simple string search command in Immediate mode was described in Chapter 4. Typing CTRL Y and then any other character moves the text pointer to immediately after the next occurrence of that character.

A more versatile string search command is available in Command mode:

Ftext$ Starting from the current position of the text pointer, search forward through the text buffer for the first occurrence of the character string 'text' and reposition the pointer immediately after it.

Thus, to find the next occurrence of Smith, the command is:

```
*FSmith$$
```

with the dollar signs standing as usual for the ESC key.

If the computer cannot find an occurrence of the specified string, it prints a message such as

```
?Can't find? FSmith$
```

showing both the nature of the error and the command that caused it. The text pointer remains where it was.

Notice that the first ESC is needed to tell the computer where the character string ends. The second ESC causes the command to be executed.

The F command can be preceded by a numerical argument which will be referred to from now on as 'n'. In the case of the F command, n can be any positive number greater than zero and tells the computer that it is the nth matching string that is wanted.

To find the third occurrence of Smith (starting from the text pointer)

 Copyright May 1980

the command is

*3FSmith$$

When no value for n is supplied, +1 is assumed. If zero or a negative value is given, the error message

?Bad argument? -1FSmith$

results.

STRING REPLACEMENT

One string of characters can be replaced by another. The command to do this is:

nXtext$ Replace n characters, starting from the text pointer, with the character string 'text'.

In the X command, n specifies the number of characters to be replaced and can be positive or negative. A negative value refers to that many characters before the text pointer, a positive value to characters after it.

Thus, assuming that the pointer has been positioned immediately after an occurrence of Smith by the F command, the command to replace Smith by Smythe is:

*-5XSmythe$$

Notice that the length of the replacement string need not be the same as that of the string it replaces.

As a convenience, typing the equals sign (=) instead of a number is equivalent to 'minus the length of the string most recently found by the F command'. So to replace the next occurrence of Smith by Smythe we could type:

*FSmith$$
*=XSmythe$$

The = is equivalent to -5 in this case.

As a further convenience, two or more commands can be grouped together into a compound command. In Command mode, execution does not occur until two ESCs are typed in succession. Thus the single command

*FSmith$=XSmythe$$

is equivalent to the previous example.

There are three ESCs in this compound command. The first two terminate the F and X strings, respectively, and the third causes the compound

 Copyright May 1980

command to be executed.

COMMAND REPETITION

We now know how to replace one occurrence of Smith by Smythe but it would still be tedious to work through a whole document issuing the same command time after time.

A command can be repeated by enclosing it in angle brackets (<>) and preceding the opening angle bracket by the number of times to carry out the repetition. The enclosed command can be simple (a single command) or compound.

The first five occurrences of Smith can be changed to Smythe by the command:

```
*5<FSmith$=XSmythe$>$$
```

Note that the number of ESCs in the command has gone up to four. The two ESCs that cause the compound command to be executed have been brought out to after the closing angle bracket but the other two that terminate the F and X strings are still needed.

By using a large number of repetitions (e.g. 999) you can ensure that all the Smiths are replaced by Smythe. The command goes on repeating until the string Smith can no longer be found.

As a further convenience, an asterisk (*) used in place of a number is equivalent to specifying +32767, the largest number TXED can cope with.

Thus to replace all the occurrences of Smith by Smythe, and assuming that you are in Immediate mode, you should move the pointer to the beginning of the text buffer by typing CTRL B, enter Command mode by typing ESC, and then enter and execute the following command:

```
**<FSmith$=XSmythe$>$$
```

where $ means ESC.

MACRO COMMANDS

There is another way of repeating a compound command. It can be stored as a 'macro' (large) command and then executed one or more times from either Command or Immediate mode. Macro commands are described in Chapters 9 and 12.

SUMMARY

Only two new commands have been mentioned in this Chapter:

nFtext$ Starting from the text pointer, search forwards for the nth occurrence of the string 'text' and reposition the pointer immediately after it.

 Copyright May 1980

nXtext$ Replace n characters, starting from the text pointer, with the text string 'text'.

The numerical argument n can is normally entered as a positive or negative number. An equals sign (=) or asterisk (*) can replace a number and represent minus the length of the last F string, or the largest positive number, respectively.

In Command mode, a compound command consists of a sequence of simple commands and is executed as a single command. Simple and compound commands can be grouped within angle brackets and executed repeatedly.

 Copyright May 1980

CHAPTER 7

INTRODUCTION TO FORMATTING

Formatting is the name given to the process of achieving a neat and attractive layout of a document on a suitable printer. The version of TXED that includes a text formatter provides a reasonable set of formatting controls, sufficient to print a document such as this Manual, and will produce output suitable for printers ranging from a teletype, through the relatively simple OKI Microline 80, to daisy wheel printers as sophisticated (and expensive) as the Qume Sprint 5. Its features include automatic right margin justification, pagination (skipping over the fold in the paper), page numbering and titling, centering, indentation, and multiple line spacing.

A formatter is an important tool for anyone who writes, because, once correct, a document is never retyped. This has the obvious benefits that the number of errors decreases with time and that the layout can be changed as often as necessary. In addition, for the occasional or inexpert typist, the satisfaction of being able to produce first class work is considerable.

The aim of this Chapter is to present an introduction to text formatting by means of an illustrative example. After reading this Chapter you should go on to Chapter 11 which defines the general concepts involved as well as giving detailed descriptions of all the formatting commands. The commands are summarised in the Quick Reference Guide in Appendix A.

GETTING STARTED

In the descriptions that follow it is assumed that you will be working at the computer, that a printer is available, and that a printer option has been selected. The 380Z computer supports various types of printer and the user must inform the computer which sort is attached. If no printer option is selected, output intended for the printer is sent to the screen.

A number of ways of selecting a printer option are available. Connecting and setting up a printer on a new system can be difficult, particularly if the printer was not supplied by Research Machines. The selection of a printer option is described in Appendix F. If in difficulty, seek the help of a more experienced colleague, or refer to the data sheet that accompanied the printer when it was supplied.

You should select a printer option now, then load TXED into the computer and type in a letter such as the one shown in Figure 7.1. You will probably find it helpful to copy it fairly exactly.

 Copyright May 1980

```
Messrs. Wright & Co.,
126B Hay Walk,
Islip,
Oxford OX95 3RM.


Dear Sirs,

PRODUCTION OF NEW CATALOGUE

We shall be grateful if you will
send us a quotation for designing,
printing and binding our new catalogue.

We enclose a copy of the last
one which was prepared for us by
another firm.
We would like to keep to the same
general style but if you wish to
suggest improvements please
let us know.


Yours faithfully,
for DOYLE & JEROME LTD.




A. BLOGGS
Manager


Enc. Catalogue (2nd Edition)
```

Fig 7.1 Form of letter as typed in to the computer.

Messrs. Wright & Co., 126B Hay Walk, Islip, Oxford OX95 3RM.

Dear Sirs,

PRODUCTION OF NEW CATALOGUE

We shall be grateful if you will send us a quotation for designing, printing and binding our new catalogue.

We enclose a copy of the last one which was prepared for us by another firm. We would like to keep to the same general style but if you wish to suggest improvements please let us know.

Yours faithfully, for DOYLE & JEROME LTD.

A. BLOGGS Manager

Enc. Catalogue (2nd Edition)

Fig 7.2 Form of letter as printed by the QT command. No formatting commands have been used so far. The page number has been omitted from this figure.

 Copyright May 1980

Having typed the letter into the computer and made any necessary corrections it is good practice to save it in a file, for example by the commands:

```
*EWLETTER.TEX$$
*QX$$
*EX$$
```

These commands are described in Chapters 3 and 5. Saving the letter in a file will save you from typing it in again if anything goes wrong.

PRINTING THE CONTENTS OF THE TEXT BUFFER

Next read the letter back into the text buffer by the commands:

```
*ERLETTER.TEX$$
*R$$
```

It is possible to print the contents of the text buffer at any time by means of the P command, which is given in Command mode:

nP Output n line of text to the printer

The P command and the syntax for the number 'n' that precedes it are described in detail in Chapter 9. For the present, however, there is no need to be concerned with the details. Make sure that the text pointer is at the beginning of the text buffer (type CTRL B if it is not), then, in Command mode, type:

```
*999P$$
```

This command means 'starting from the position of the text pointer, print 999 lines of text'. The command stops if the end of the text buffer is reached before 999 lines have been printed. Specifying an arbitrarily large value for n is a convenient way of saying you mean the remainder of the text buffer.

The letter should now be printed and the printed output should be an exact copy of the text buffer contents. In particular, the lines of text are still short, just as they were when typed in.

The P command is useful for making a permanent record of the contents of the text buffer; by specifying a smaller value for n you can print selected portions from it.

FORMATTED PRINTING

The P command printed out the contents of the text buffer literally. To print the text after formatting, the QT command is used:

QT Output formatted text to printer

In contrast to the P command, no value for 'n' is needed; QT always prints the whole of the text buffer. Adjust the printer so that the print head is next to the top of a page, then type:

*QT$$

The letter will be printed again, with the result as shown in Figure 7.2. This time there have been changes. If you are using a standard printer, such as an OKI Microline 80 or Anadex DP-8000, with the common 9.5 by 12 inch perforated computer paper (approximately A4 size), the results will probably be quite reasonable. This is because the formatter assumes initial values for parameters such as line length and page length and has been set up to suit such printers. There are of course commands to change these parameters and in addition the initial values can also be changed.

If this first attempt at formatting produces completely unacceptable results, you may want to refer to Chapter 11, and in particular, to Figure 11.1 which illustrates the commands used for laying out a page.

Assuming that what was printed was similar to Figure 7.2, the points to notice are that the letter is now neatly centered within the page and that the length of each line of the second paragraph is the same. A page number has been printed at the bottom of the page.

The centering of the letter within the page is due to the assumed values for the width of the left margin and line length. In the second paragraph, an even right margin has been achieved firstly by moving words up from subsequent lines, and then, when no more words will fit, by adding spaces to bring the line to the correct length. These two processes are known respectively as filling (of lines with words) and justification (of the right margin). Filling with justification is the initial mode of the formatter, since it is what most people want to do, but commands are also available to output text literally and to perform filling without justification.

FORMATTING COMMANDS

The appearance of the letter after formatting is fair but some aspects are worse than the original, examples being the address and salutation where the lines have been filled unnecessarily. The formatter is designed to require few commands to tell it what to do but it does need some guidance.

A formatting command consists of a dot (full stop), a two-letter command and perhaps some other information. Each command must appear at the beginning of a line with nothing else on that line.

A full stop at the beginning of a line is a rare occurrence in normal text and should be avoided in text to be formatted except when you want to give a formatting command. Note that the full stop must be in the first character position of the line for it to be recognised as the start

of a command. Even one space before the full stop will cause the computer to treat the command as text.

The format commands have an effect on the mechanical printer similar to the marks and symbols inserted by a proof reader to tell the human printer how the type is to be set.

The first command we need is one to tell the formatter not to fill lines:

.LI Output text literally

We also need one to cause filling and justification to resume:

.FJ Output text after filling and justification

These commands can be placed around the lines we want to be printed without filling, for example:

```
.LI
Messrs. Wright & Co.,
126B Hay Walk,
Islip,
Oxford OX95 3RM.
.FJ
```

Further commands that will improve the layout of the letter are:

.CE Centre the following line

.PP Begin a new paragraph

.INn Indent by n character positions

.CI Cancel indentation

All the commands can be given in upper or lower case, that is, the two forms .CI and .ci have an identical effect. Where a number is expected, as in the .IN command, the number may immediately follow the command letters or be separated from them by one or more spaces.

Formatting commands should now be inserted in the relevant places in the text, as shown in Figure 7.3.

The result of repeating the QT command is shown in Figure 7.4.

```
.LI
Messrs. Wright & Co.,
126B Hay Walk,
Islip,
Oxford OX95 3RM.
.FJ


Dear Sirs,

.CE
PRODUCTION OF NEW CATALOGUE
.PP
We shall be grateful if you will
send us a quotation for designing,
printing and binding our new catalogue.
.PP
We enclose a copy of the last
one which was prepared for us by
another firm.
We would like to keep to the same
general style but if you wish to
suggest improvements please
let us know.


.IN 17
.LI
Yours faithfully,
for DOYLE & JEROME LTD.




A. BLOGGS
Manager
.CI


Enc. Catalogue (2nd Edition)
```

Fig 7.3 Form of letter in text buffer after formatting commands have been added.

Messrs. Wright & Co.,
126B Hay Walk,
Islip,
Oxford OX95 3RM.

Dear Sirs,

PRODUCTION OF NEW CATALOGUE

We shall be grateful if you will send us a quotation for designing, printing and binding our new catalogue.

We enclose a copy of the last one which was prepared for us by another firm. We would like to keep to the same general style but if you wish to suggest improvements please let us know.

Yours faithfully,
for DOYLE & JEROME LTD.

A. BLOGGS
Manager

Enc. Catalogue (2nd Edition)

Fig 7.4 Form of letter as printed by the QT command after formatting commands have been inserted. As in Fig 7.2, the page number has been omitted.

GENERAL OBSERVATIONS

The example letter as shown in Figure 7.4 now looks fairly neat although how neat you think it is will depend on how you like your letters set out. This Manual does not pretend to give an account of conventions for laying out documents and the less experienced reader should consult a book such as Pitman College Typewriting (Teach Yourself Series) 1974.

Only a few of the formatting commands have been mentioned in this Chapter and for a full account of the available commands you must refer to Chapter 11.

In particular we have not yet mentioned the commands for setting up the margins, line and page length, running title, and folio (the running footing at the bottom of the page). These are usually set up by a series of formatting commands at the beginning of a document and not changed thereafter.

Once you have decided on a layout you like, it is both timesaving and useful to save this sequence of commands in a file. The standard set of commands can then be read into the text buffer before typing in a document.

In addition to the QT command, which is intended for continuous stationery, the QP command is available for use with single sheets. The QP command pauses at the end of each page to allow a new sheet to be inserted.

By giving the QW command, formatted output can be sent to a file and can be used for the preparation of standard letters. An example is given of this in Chapter 12.

The QV command is particularly useful if you have an 80 character screen, although it can be used with 40 character screens. This command sends the formatted text to the screen. TXED will also do on-screen underlining and bold face if you have an 80 character screen. See Chapter 9 for more details.

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

CHAPTER 8

IMMEDIATE MODE REFERENCE SECTION

This Chapter is the first of the four Reference Chapters in the Manual and is concerned with the Immediate mode commands. Subsequent Chapters cover Command mode and Formatting.

Descriptions of the majority of the Immediate mode commands have already been given in Chapters 2 and 4 and will not be repeated here. The main purpose of this Chapter is to list all the commands available and to define how some of them work. As mentioned at the end of Chapter 2, it is possible for the user to change both the control keys that invoke the commands and the commands themselves. If you wish to do this, you will need a good knowledge of Command mode, described in Chapter 9, and will have to refer also to Appendix E.

IMMEDIATE MODE COMMANDS

All the available control keys are listed in this section. Most of them initiate an Immediate mode editing command. Unless directed elsewhere, refer to Chapters 2 and 4 for further details.

CTRL A Turns on and off 'autopaging' of the screen. This is a COS Monitor command and is only useful in editing in conjunction with the Command mode T or QV commands or when using one of the print commands without selection of a printer option. These commands are described in Chapter 9.

When autopaging is turned on, output to the screen pauses after the screen has filled. Typing any character except CTRL A causes the next screen full to be output. Typing CTRL A once turns off autopaging; typing it a second time turns autopaging on again.

You will only be able to see the bottom four lines of the output to the screen when the text window is displayed. The text window can be disabled by the 0ED command, described in Chapter 9.

CTRL B Move text pointer to beginning of text.

CTRL C Return immediately to Operating System (see Chapter 5).

CTRL D Move text pointer forward one character.

CTRL E Delete character after text pointer.

 Copyright May 1980

CTRL F Call Front Panel. This is a COS Monitor command and is mainly useful in editing for selecting a printer option when you have forgotten to do so before loading TXED (see Appendix F) and for recovering from an error (see Appendix B).

Because it is annoying to type CTRL F by accident, you are asked:

Front Panel check (Y/N)?

Type Y or y if you really want to call up the Front Panel; typing any other character will abort the CTRL F command. Once you have entered the Front Panel you can return to TXED by typing K.

CTRL G Move text pointer backwards one line.

CTRL H Not used.

CTRL I Insert TAB character. TAB characters are used to produce a neat layout in computer programs, expecially those written in Assembly Language. Refer also to the description of the ET command in the Utility Commands section of Chapter 9.

CTRL J Insert blank line after text pointer.

CTRL K Delete line after text pointer.

CTRL L Insert FORM FEED character. FORM FEED characters can be used to divide up a piece of text into two or more logical 'pages'. When the text is read into the text buffer from a file, reading will stop on encountering each FORM FEED character and the warning message:

More on input

will be printed. The FORM FEED character will have been included in the text buffer and is shown in the text window as ↑L (uparrow L). Editing such a file is exactly like editing a large file, except that the sections are shorter, and involves use of the Command mode N command. Refer to Chapters 4 and 10 for further details.

CTRL M Same as RETURN key.

CTRL N Move text pointer forwards one line.

CTRL O Insert text marker before the text pointer (CTRL Y ESC finds next text marker).

CTRL P Move text pointer forwards one page.

CTRL Q Move text pointer backwards one page.

CTRL R Delete word before text pointer.

CTRL S Swap case of letter from upper to lower or vice versa.

CTRL T Not used.

CTRL U Move text pointer backwards one word.

CTRL V Move text pointer backwards one character.

CTRL W Move text pointer forwards one word.

CTRL X Execute Macro command once. This very useful command is described in Chapter 12.

CTRL Yc Move text pointer forwards to just after character c, where c is any character.

CTRL Z Move text pointer to end of text.

CTRL [Same as ESC key. 27

CTRL \ Not used. 28

CTRL] Not used. 29

CTRL ↑c Insert character c as control character. The main use of this command is to allow an ESC character to be inserted into the text buffer. This is sometimes needed when editing a macro command (see Chapters 9 and 12). To insert an ESC character, type CTRL ↑ ESC.

 Copyright May 1980

CORRESPONDENCE BETWEEN IMMEDIATE AND COMMAND MODES

The Immediate mode editing commands are in fact defined in terms of Command mode editing commands. It is possible to redefine most of the Immediate mode commands if you want to. Refer to Appendix E for details. The following table lists the current definitions of the commands:

Immediate mode	Command mode	Description of command
CTRL B	B	Move text pointer to beginning of text
CTRL D	+1C	Move text pointer forward one character
CTRL E	+1D	Delete character after text pointer
CTRL G	-1C0L	Move text pointer backwards one line
CTRL J	IRETURN$-1C	Insert blank line after text pointer
CTRL K	+1K	Delete line after text pointer
CTRL N	+1L	Move text pointer forwards one line
CTRL O	I$	Insert text marker before the text pointer
CTRL P	+15L	Move text pointer forwards one page
CTRL Q	-15L	Move text pointer backwards one page
CTRL R	-Y	Delete word before text pointer
CTRL S	O	Swap case of letter
CTRL U	-U	Move text pointer backwards one word
CTRL V	-1C	Move text pointer backwards one character
CTRL W	+U+U-U	Move text pointer forwards one word
CTRL X	1EM	Execute Macro command once
CTRL Yc	1Fc$	Find character c
CTRL Z	Z	Move text pointer to end of text
CTRL ↑c	↑c	Insert character c as control character
DELT	-1D	Delete character before text pointer

CHAPTER 9

COMMAND MODE REFERENCE SECTION
PART 1: EDITING COMMANDS

In this Chapter we begin the formal description of Command mode. The Chapter is divided roughly into two parts, the first describing how a command is constructed and the second being the individual descriptions of the commands. The Command mode Reference Section continues in the next Chapter with a description of the commands involved in the handling of files.

COMMAND TERMINATION AND EXECUTION

TXED is in Command mode when an asterisk prompt (*) appears at the beginning of the bottom line of the screen. In Command mode, a command is not executed until two ESC characters are typed in succession. You will already be familiar with the simplest form of command in Command mode, viz. two ESCs typed in succession at the beginning of a line. This command causes TXED to enter Immediate mode.

The ESC character is displayed on the command line as a dollar sign.

A clear distinction must be drawn between the two separate uses of ESC in Command mode. A single ESC terminates a text string argument (see below). Two ESC characters typed in succession terminate a series of one or more commands and cause them to be executed. Thus the command:

```
ERB:TEST1.TEX$
```

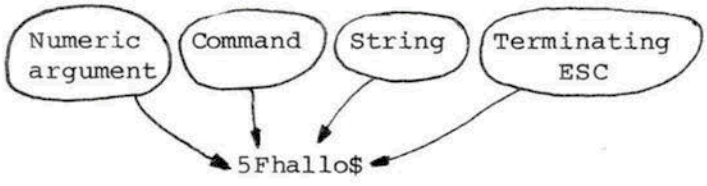
is complete in itself but is not executed until a second ESC is typed. You could choose instead to give the command in the form:

```
ERB:TEST1.TEX$R$$
```

which not only opens the file B:TEST1.TEX for reading, but also reads it into the text buffer. This second form is called a 'compound command' (see below).

COMMAND STRUCTURE

A command consists of up to four parts: a numeric argument, a one or two letter command, a text string, and a terminating ESC character. A typical command is:



 Copyright May 1980

The numeric argument consists of a sign (+ or -) and a decimal number, both of which are optional. If omitted, the sign defaults to positive and the number to 1. The number must be in the range 0 to 32767. In the descriptions of the commands, the numeric argument is often referred to simply as 'n'.

The command itself consists of one or two letters and is always present. Commands may be entered in upper or lower case.

The text string is optional, depending on the command, and consists of a sequence of characters, for example, a file specification or a word that you wish to search for in the text buffer.

If such a string is present, it must be followed by a terminating ESC character.

Many commands require fewer than four parts. For example, the B command (move pointer to beginning of the text buffer) requires that its numeric argument is +1 and that it is not followed by a text string. While it could be typed as +1B, +B or 1B, it is normally given simply as B.

On the other hand, the F command (find nth matching text string) requires both numeric and text string arguments, the latter terminated by an ESC character. The numeric argument can be left out (the default of +1 is taken) and the text string may be empty (i.e. contain no characters) but the terminating ESC character must always be present.

Some examples of commands are:

B	Move to beginning
5c	Move forward 5 characters
-10K	Delete previous 10 lines
Fhallo$	Find next 'hallo'
3fhallo$	Find third 'hallo'
F$	Find next text marker

None of the above commands would of course be executed until two ESC characters are typed in succession.

SPECIAL FORMS OF THE NUMERIC ARGUMENT

As a convenience, an asterisk (*) or an equals sign (=) may be used in place of the numeric argument.

An asterisk is equivalent to typing the largest number that TXED recognises (32767) and may be preceded by a plus or minus sign. It is useful for performing a command an arbitrarily large number of times. For example, the command:

```
*K
```

deletes the text buffer contents from the pointer to the end of the buffer, and:

```
-*K
```

 Copyright May 1980

deletes all text from the beginning of the buffer to the pointer.

The equals sign is equivalent to minus the length of the string argument of the most recent F command. Its main purpose is to simplify the X command in 'search and replace' operations. Refer to the description of the X command (below) for examples.

COMPOUND COMMANDS

As mentioned earlier, a command is not executed until two ESC characters are typed in succession. This allows a series of commands (referred to from now on as 'simple' commands) to be combined into a compound command before execution. For example, the compound command:

```
B*K$$
```

moves the pointer to the beginning of the text buffer, then deletes all the lines in the buffer (i.e. it clears the buffer).

If an error occurs during the execution of a (simple) command within a compound command, the remaining (simple) commands are not executed.

Compound commands are particularly useful in constructing macro commands (described below and in Chapter 12) and when using iteration brackets (see next section).

ITERATION

It is sometimes desirable to be able to repeat a command several times. This may merely save the effort of typing it repeatedly, or it may save you remaining at the keyboard while TXED is formatting a long document.

A command may be repeated by enclosing it within angle brackets (<>). These are referred to as iteration brackets. The opening angle bracket is preceded by a numeric argument specifying the number of times to repeat the group of commands within the brackets.

For example, the command:

```
20<I-$>$$
```

would insert a string of 20 dashes into the text buffer at the pointer position and saves typing (and counting) 20 dashes.

The more complex command:

```
ERMYBOOK.TEX$*<RQT*K>$$
```

would read, format and print the contents of the file MYBOOK.TEX without intervention, even if the file was long enough to fill the text buffer several times over (see Chapter 11).

A bracketed group of commands may itself be enclosed within iteration brackets, up to a maximum nesting level of eight. Commands within a

group are executed from left to right.

SPECIAL CHARACTERS

The characters ESC, asterisk (*), equals (=), and angle brackets (<>), which have special significance in Command mode, have already been mentioned. Other such characters are:

RETURN — May be used within the text string of an I, F or X command, or to construct a multi-line command. Except when included within a text string, RETURN characters are ignored in Command mode and are used merely to improve the readability of complex compound commands. In particular, the RETURN character does not act as a command terminator.

SPACE — Space characters are ignored within commands, except within string arguments.

DELT — Deletes the last character typed. In a multi-line command, DELT may be used to delete past the beginning of one line and into the previous line. If all the characters in a command are deleted, the asterisk prompt is repeated. If you make a mistake in typing the second of two ESCs and delete that second character, you will have to type two further ESCs to cause your command to be executed.

CTRL C — Return to Operating System (see Chapter 5).

CTRL F — Call Front Panel (see Chapter 8).

CTRL R — Retype the whole of the current command line. This is useful for redisplaying a multi-line command when part of it has been edited by CTRL U.

CTRL U — Delete the command line back as far as the previous RETURN character. CTRL U is useful for deleting the last line of a multi-line command.

CTRL X — Delete the whole of the current command line. The asterisk prompt is repeated.

COMMANDS THAT MOVE THE TEXT POINTER

The commands in this group move the text pointer varying distances within the text buffer.

B — Move pointer to beginning of text buffer

The B command moves the text pointer to the beginning of the text buffer, before the first character. It is useful for going back to the start of the text before searching for a section of text to be edited.

The B command ignores any numeric argument given.

nC Move pointer by a specified number of characters

The C command moves the text pointer by a given number of character positions. If the numeric argument n is negative, the pointer is moved back n characters; if it is positive, the pointer is moved forward n characters. If the argument is zero the pointer is not moved.

If the length of the move would take the pointer past the beginning or end of the buffer it is moved only to its beginning or end.

The nC command requires that space, TAB and RETURN characters be included in the value of n.

nL Move pointer by a specified number of lines

The L command moves the text pointer to the beginning of a line of text. If the numeric argument n is positive, the pointer is moved to the beginning of the nth line after the present line. If n is zero, the pointer is moved to the beginning of the line in which it lies. If n is negative, the pointer is moved to the beginning of the nth line before the present line.

If the length of the move would take the pointer past the beginning or end of the buffer it is moved only to its beginning or end.

nU Move pointer by one word

The U command moves the text pointer forward or backwards by one word. The numeric argument n must be plus or minus one. Two forms of word definition are available, selectable by the QU command (see below, Utility Commands), which cause the pointer to come to rest at the beginning or end of a word.

If the search for the beginning or end of a word would take the pointer past the beginning or end of the buffer it is moved only to its beginning or end.

Z Move pointer to end of text buffer

The Z command moves the text pointer to the end of the text buffer, after the last character. It is useful for going to the end of the text before inserting more.

The Z command ignores any numeric argument given.

 Copyright May 1980

COMMANDS THAT CAUSE DELETION OF TEXT

The commands in this group move the text pointer within the text buffer and at the same time delete the text between the old and new positions.

nD Move pointer with deletion of characters

The D command moves the text pointer by the specified number of characters and deletes the text between the original text pointer and the new position.

If the numeric argument n is negative, the pointer is moved back n characters; if it is positive, the pointer is moved forward n characters. If the argument is zero the pointer is not moved.

If the length of the move would take the pointer past the beginning or end of the buffer it is moved only to its beginning or end.

Refer to the description of the X command below for an example of the use of the D command in a search and delete operation.

nK Move pointer with deletion of lines

The K command moves the text pointer by the specified number of lines and then deletes the text between the current text pointer position and the new position.

If the numeric argument n is positive, the pointer is moved to the beginning of the nth line after the present line. If n is zero, the pointer is moved to the beginning of the line in which it lies. If n is negative, the pointer is moved to the beginning of the nth line before the present line.

If the length of the move would take the pointer past the beginning or end of the buffer it is moved only to its beginning or end.

nY Move pointer with deletion of one word

The Y command moves the text pointer by one word and then deletes both the word and the non-alphanumeric characters that delimit it. The numeric argument n must be plus or minus one.

The number and position of the characters that are deleted depends upon the word definition that is currently in force. The definition of a word can be changed by the QU command (see below, Utility Commands),

 Copyright May 1980

TEXT MODIFICATION COMMAND

O Move pointer with case change

The O command moves the text pointer forward one character position. The numeric argument n must be plus one. If the character over which the pointer passes is an upper case letter (A to Z) it is changed to the corresponding lower case letter (a to z). If the character is in lower case it is changed to upper case. If the character is not alphabetic it is not changed.

TEXT INSERTION

Itext$ Insertion of new text

The I command inserts a text string into the text buffer at the position of the text pointer. The text string may consist of a number of lines of text. The TAB command (CTRL I) is similar to the I command except that the TAB character is also counted as part of the text string and is inserted into the text buffer before the text string.

After insertion the text pointer is positioned after the last character inserted. The I and TAB commands must have numeric arguments of +1.

The I command is normally used only for inserting small amounts of text, such as one or two words. The maximum length of the text string that can be inserted is around 250 characters. Use Immediate mode for major inserts.

If there is not enough room within the text buffer to accept the text string, then the string will not be inserted and the command will be aborted with the error message

```
?No room?
```

Note that the command I$ without a text string inserts a text marker into the buffer at the position of the text pointer.

↑C Insert control character

The uparrow command converts the character which follows it (usually an upper case letter) into a control character and inserts it into the text buffer immediately before the text pointer. The numeric argument must be +1.

 Copyright May 1980

SEARCH AND REPLACEMENT OF TEXT

The two commands in this group support string search with optional deletion or replacement.

nFtext$ Find text string

The F command searches the text buffer to find a portion of text which matches the text string contained within the command. Starting from the position of the text pointer, the command searches forward for the nth matching string. If the search is successful, the pointer is moved to a new position immediately after the matching string. If the search is unsuccessful the pointer is not moved and the command string is aborted with the message

```
?Can't find? nFtext$
```

where 'text' is the string that cannot be found. The numeric argument n must be greater than zero.

If the user wishes to find the last occurrence of a matching text string, the F command can be used within iteration brackets. When the match fails the pointer will be left immediately after the last successful match. Refer to the beginning of this Chapter for further details of iteration brackets.

Note that the command nF$ without a text string will find the nth text marker following the text pointer position and place the pointer after it.

nXtext$ Replace n characters with text string

The X command causes deletion of n characters (counting space and RETURN characters) and replaces them with the text string contained within the command.

The number of characters to be deleted in the exchange is evaluated in the same way as in the D command. In fact the nXtext$ command is functionally the same as nDItext$.

The same rules apply to the text string as for the I command, and the X command will also produce an error if there is not enough room for the replacement text to fit into the text buffer.

An equals sign (=) may be used in place of n and is equivalent to minus the length of the last string given in a Find command. The equals sign is often useful in search and delete and search and replace operations. For example:

```
FRubbish$=D$$
```

 Copyright May 1980

will search for the string 'Rubbish' and delete it, and:

```
FEna Sharples$=XRaquel Welch$$
```

will find and replace 'Ena Sharples' with 'Raquel Welch'.

COMMANDS INVOLVING TEXT MARKERS

A text marker is inserted into the text buffer by an I command with an empty text string (I$$). The marker is inserted at the position of the text pointer. The X command with an empty text string has a similar effect (nX$$). The F command with an empty text string finds the nth text marker, searching forward from the text pointer position (nF$$).

Text markers have three functions:

As place markers
For delimiting blocks of text
To allow selective output during formatting

When used as place markers, markers are usually inserted and found by the Immediate mode commands CTRL O and CTRL Y ESC. When blocks of text are involved, the pairs of markers that delimit a block are usually inserted in Immediate mode and then one of the commands in this group is executed. Use of markers in text formatting is covered in Chapter 11.

An introduction to the use of text markers will be found in Chapter 4.

QC Copy marked text

The QC command copies a block of text delimited by two markers, inserting the copy immediately before the text pointer. The text pointer must be outside the marked block (see QM command, below). Because this command generates extra text there must be enough room in the text buffer to hold it, otherwise the error message

```
?No room?
```

will be displayed. The markers are not deleted.

nQD Delete text markers

The QD command deletes all the text markers in the selected portion of the text buffer. If the numeric argument n is positive, all text markers from the text pointer to the end of the buffer are deleted. If n is negative, all markers from the beginning of the buffer to the text pointer position are deleted. A zero value of n will cause all markers to be deleted, both before and after the text pointer.

Text markers may also be deleted like any other character, using the D and K commands, or the Immediate mode commands

 Copyright May 1980

CTRL E, CTRL K and DELT.

QK Delete marked text

The QK command deletes a block of text between two markers and deletes the markers.

QM Move marked text

The QM command moves the block of text delimited by two markers to a position immediately before the text pointer. The pointer must be before or after both markers, not between them, otherwise the error message

?Markers?

will be displayed and no move will take place. The markers are deleted after a successful move.

It should be noted that the QC, QK and QM commands are applied to the first delimited block of text found, starting from the beginning of the text buffer.

All text markers are removed when text is written to a file. They will no longer be present when that text is read back into the text buffer.

OUTPUT OF TEXT TO SCREEN OR PRINTER

This group of commands is used to output the contents of the text buffer to the screen or printer.

If no printer option has been selected, the output intended for the printer is sent to the screen. Selection of a printer option is described in Appendix F.

The T and P commands are used to output selected portions of the text buffer. The text is output more or less unchanged, the exceptions being that a LINE FEED is added after each RETURN character, TAB characters are expanded according to the current setting of the ET command (see below, Utility Commands), and control characters are output in the form of an uparrow followed by the corresponding upper case letter. Formatting commands embedded within the text are not processed and are output literally.

The QP and QT commands output the whole of the text buffer, formatting its contents before sending it to the printer. The QV command sends formatted output to the screen. Formatting commands embedded within the text define how the text should be laid out and are not printed.

All the commands (except QW) can be interrupted by typing CTRL Q.

nT Output n lines of text to screen

The T command displays lines of text on the console screen. The beginning of a line of text is found in the same way as for the L and K commands (q.v.). Then the text between the text pointer and the start of that line, or between the start of that line and the pointer, is printed on the screen. The text pointer is not moved and the text in the buffer is not changed.

The text is displayed on the bottom four lines of the screen if the text window is active, or on the whole screen if it has been disabled. The text window is enabled or disabled by the ED command (see below, Utility Commands).

If n is zero, the text from the beginning of the current line up to the text pointer is printed. A numeric argument of +1 prints the text from the pointer to the end of the current line. The command 0T followed by 1T will print the whole of the line containing the text pointer without a break.

If a line of text typed by the T command extends beyond the right hand edge of the screen it is automatically continued on the next line. This feature can be used to display long lines of text.

When the text window is disabled the T command is the principal way of viewing the contents of the text buffer.

nP Output of n lines of text to printer

The P command is similar to the T command except that its output goes to the printer, not the screen.

QT Output formatted text to printer

The QT command moves the text pointer to the beginning of the text buffer, then formats the text according to the formatting commands contained within it. The formatted text is sent to the printer.

This command outputs the whole of the text without stopping and is intended for use with continuous or fan-fold paper.

QP Output formatted text to printer with pause

The QP command is similar to the QT command except that output to the printer stops at the end of each page to allow a new sheet of paper to be inserted. Type RETURN to resume printing.

 Copyright May 1980

QV Output formatted text to screen

The QV command is similar to the QT command except that the formatted text goes to the screen. The left margin (set by the .LM command described in Chapter 11) is set to zero and the special daisy-wheel sequences are suppressed, similar to issuing a .QN command (also described in Chapter 11). However, on an 80 character board, bold face and underline commands are interpreted to produce dim and underlined characters respectively.

nQZ Output CTRL Z to printer

The QZ command sends n CTRLZ characters to the printer device. The main use for this command is on network systems, where the CTRL Z character is used as a signal to the print spooler to start printing. This command is not normally used on stand-alone systems.

The QW command is similar to the QT and QP commands, except that the formatted text is sent to a file. It is described in Chapter 10.

Side effects of the QT, QP, QV and QW commands are that the text pointer is moved to the beginning of the text buffer and that the contents of the Macro/Save buffer are lost.

Note that with the QT, QP, QV and QW commands, text markers can be used to output selected portions of the text buffer. See Chapter 11.

SAVING AND COPYING SMALL PORTIONS OF TEXT

The two commands in this group may be used to move around or duplicate small portions of text. Text is saved by the S command and the saved text copied back into the text buffer by the G command.

nS Save n lines of text in Macro/Save buffer

The S command saves a copy of n lines of text in the Macro/Save buffer. A new text pointer position is calculated in the same way as for the L and K commands (q.v.). The numeric argument n may be positive or negative. The command then copies the text between the calculated position and the current text pointer into the Macro/Save buffer. Neither the position of the text pointer nor the text in the text buffer is changed. The previous contents of the Macro/Save buffer are lost.

The amount of text to be saved cannot exceed the capacity of the Macro/Save buffer (512 characters). It will hold about 6 to 20 lines of text, depending on length. If the command would overfill the buffer, the error message

?No room?

is displayed and the command is aborted.

G Insert text from Macro/Save buffer

The G command inserts a copy of the text saved in the Macro/Save buffer into the text buffer immediately before the text pointer. The text pointer is positioned at the end of the string so copied. The numeric argument n must be +1.

If there is insufficient room within the text buffer to insert the text, the text pointer is not moved and the command is aborted with the error message

?No room?

The G command does not affect the contents of the Macro/Save buffer and can therefore be repeated several times. To insert multiple copies, the G command may be enclosed within iteration brackets (see the beginning of this Chapter).

It should be noted that the Macro/Save buffer is used for three purposes:

Saving text
Holding a macro (see below)
For control functions during formatting

It can only be used for one of these purposes at a time, and neither the S nor the G command is allowed within a macro command.

When using the S and G commands to move a block of text, it is convenient to follow the nS command by an nK command with the same value of n. The command sequence SKLG is also useful for interchanging two lines.

Where more than a few lines of text have to be copied, it is often more convenient to mark the block with text markers and use one of the block move commands described in Chapter 4.

SAVING AND EXECUTING A MACRO COMMAND

The two commands in this group allow a simple or compound command to be saved in the Macro/Save buffer for subsequent execution. The M command saves the remainder of the command line. The nEM command executes it one or more times. Macro commands are considered in more detail in Chapter 12.

Mcommand$ Save command in Macro/Save buffer

The M command saves the remainder of the command line in the Macro/Save buffer. The whole of the rest of the command line is saved, up to the first of the two terminating ESC characters. The previous contents of the Macro/Save buffer are lost.

The numeric argument n must be +1. An M command cannot contain M, S, G or EM commands, nor can an M command occur within iteration brackets. Iteration brackets are, however, allowed within a macro. Thus the commands

```
MSKLG$$
5<MI $>$$
```

are both illegal, but the command

```
M5<I $>$$
```

is allowed and defines a macro to insert five space characters.

nEM Execute macro command n times

The EM command executes the contents of the Macro/Save buffer as a command. If the EM command occurs in the middle of a compound command, the command is continued after the macro has been executed. The numeric argument n must be greater than zero and determines the number of times the macro should be executed.

As pointed out in the description of the S and G commands above, the Macro/Save buffer is used for more than one purpose, and it is its doubling as a Macro and a Save buffer that prevents the inclusion of the S and G commands within a macro. The scheme does however have the advantage that a macro command can be copied into the text buffer (by the G command), edited, then saved back as a macro (by the nS command). It also provides a means whereby complex macros can be written to and read back from files.

UTILITY COMMANDS

The commands in this group are used to adjust parameters and switches that affect the behaviour of the editor. In many cases, the user can arrange that these settings are in the preferred position when TXED is loaded (see Appendix E).

nED Enable or disable text window

The ED command enables the text window if the numeric argument n is non-zero, or disables it if the argument is zero. Immediate mode editing may only be carried out when the text window is enabled.

The QL (disc version) and R (cassette version) commands automatically disable the text window.

nEL Modify text window display

The EL command allows the user to alter the behaviour of the text window.

If the numeric argument n is zero, only the first 40 (or 80) characters of a line are displayed. An exception occurs in the case of the line containing the text pointer. As the cursor indicating the position of the text pointer nears the right hand edge of the screen, the line will be displaced by a multiple of 8 spaces to the left, keeping the cursor on the screen. A graphics symbol is displayed at the left hand edge of the cursor line to indicate that the line has been shifted. This setting (0EL) is usually the most convenient for editing line oriented text such as programs and is the default setting.

If n is positive, the whole of each line is displayed. In this case, long lines are continued across two or more screen lines, with a special graphics symbol at the beginning of each continuation line. This setting (EL) is useful for seeing all the text, particularly when the lines are longer than 40 (or 80) characters. It is, however, assumed that lines are of reasonable length. Excessively long lines cannot be displayed and cause the display to revert automatically to the short line mode (0EL) with the warning message

?Display error?

Even the short line mode cannot display more than about 240 characters on the line containing the text pointer. To allow the whole of the line to be displayed, a RETURN character should be inserted before this limit is reached.

If n is negative, the behaviour of the window is initially similar to short line mode (0ED), but as the cursor approaches the right hand side of the screen, the whole of the display is shifted to the left, not just the line containing the cursor. Values for n of -1, -2, -4, -8 or -16 cause the display to be moved in multiples of 1, 2, 4, 8 or 16 spaces. This mode is particularly convenient for the construction of tables.

A further refinement is that an automatic right margin can be set by the QS command (see below) that will cause TXED to begin a new line automatically after the line being entered reaches a predetermined length. This is useful for touch typists.

nET Set tab stops to n spaces

The ET command alters the amount by which TAB characters are expanded. Tab stops are set every n character positions along a line, counting from zero, and TAB characters are expanded to a maximum of n spaces. The allowable values of n are 0, 1, 2, 4 or 8. If n is 1, TAB characters are indistinguishable from spaces. If n is zero, TAB characters become visible in the text window, being displayed as an upside down T, but are still expanded to one space by the T and P commands.

 Copyright May 1980

A TAB character is generated from the keyboard by typing CTRL I and can be inserted into the text buffer in Immediate mode, or by inclusion in within the text string of an I or X command.

TABs are used frequently to neaten program layout. In Assembly Language the default setting of 8 is usually appropriate, but in block structured languages such as Algol and C, a tab setting of 2 or 4 will often allow more of the program to be seen on the screen.

It should be emphasised that TABs are only expanded when displayed in the text window, or output by means of the T, P or formatting commands. Within the text buffer they remain as TAB characters and are written to and read from files without alteration.

nES Set screen width

The ES command can be used on systems with an 80 character board to select either a 40 or an 80 character screen. If n is positive, then 80 character mode is selected. If n is zero, then 40 character mode is selected. On a 40 character board, neither of these commands has any effect except for a slight flash on the screen.

A negative value of n is used only for configuring TXED. On initial entry, TXED selects the screen width set up by any saved ES command. By default, TXED uses the screen width already in effect, and -ES restores this state. See Appendix E for details.

nQF Set cursor flash rate

The QF command determines the rate at which the cursor that indicates the position of the text pointer flashes. The larger n is the slower the flash rate.

If n is zero, the cursor reverts to the displacement form used in earlier versions of TXED. In this form it occupies one character position rather than overlying the next character.

nQS Set automatic right margin

The QS command sets an automatic right margin. The numeric argument n must lie in the range 20 to 200.

The automatic right margin applies only during text entry in Immediate mode. When the cursor position on the line of text being entered exceeds the right margin setting, the next space character to be typed is converted to a RETURN character, forcing text entry onto a new line.

The effect of this is to restrict the entered lines to a length not much greater than n, but at the same time, the lines are

broken at the end of a word.

A setting of 32 ensures that most lines will fit on a 40 character screen, and a setting of 70 is appropriate for an 80 character screen. The default setting of 200 normally has no effect.

The QS command is intended to allow touch typists to produce text that can be edited easily without having to worry about 'going off' the right hand edge of the screen. The automatic margin should normally be disabled by setting n to 200 when entering programs.

nQU Define word

A word is defined as a group of one or more alphanumeric characters, delimited on either side by one or more non-alphanumeric characters. Two forms of definition are available.

With the first form (set by default, or after the command QU) a word move will extend over delimiters, the word itself and then any further delimiters. The action is symmetric, whether going forwards or back.

With the second form (set by the command 0QU) a word move stops after the word itself.

The differences between the two forms of definition can be explored in Immediate mode and the reader is left to decide his or her own preference.

QX Continue after EX

Normally the EX command (described in Chapter 10) returns to the Operating System after file operations are complete. If the QX command is issued prior to giving an EX command, the return to the Operating System is disabled and TXED is restarted. The QX command need only be given once.

QX avoids the inconvenience of continually reloading the text editor if many short files have to be edited.

Q* Overwrite formatter

The formatting section of the TXED program occupies about 2K bytes of memory. If you are short of space and do not intend to use the formatting instructions, giving the Q* command will delete the formatter and reuse that memory to enlarge the text buffer.

After the Q* command, the commands Q*, QW, QV, QT and QP are no longer recognised.

 Copyright May 1980

CAUTION: the Q* command totally reorganises TXED's internal buffers and then performs a cold start. The contents of the text and Macro/Save buffers are lost. If given, it should be the first command issued, and because of its potentially disastrous effect, a warning message appears. You will not be able to format text without reloading TXED.

 Copyright May 1980

CHAPTER 10

COMMAND MODE REFERENCE SECTION
PART 2: FILE RELATED COMMANDS

The Command mode Reference Section is continued in this Chapter with a description of the commands that are concerned with files. The Chapter is divided roughly into three parts. The first of these defines the form of a file specification, enlarging slightly on the description presented in Chapter 3. This is followed by the descriptions of the individual commands. Finally, a series of 'recipes' for the more common file operations is given.

FILE SPECIFICATIONS

In Chapter 3 we described how a file specification consists of three parts, the disc name, the file name, and the file type.

TXED imposes the restriction that the characters from which these three parts are constructed must consist only of letters, numbers and the dollar sign, drawn from the sets A-Z, a-z, 0-9 and $. When a file specification is processed, lower case letters are converted to the corresponding upper case ones.

The disc name consists of a single letter, usually in the range A to D, followed by a colon (:). The range of disc units available on a particular system may be smaller or greater than this. Consult your System Manual if in doubt.

If the disc name is omitted, TXED assumes that the file is on the 'logged in' disc, i.e. the disc whose name appeared at the beginning of the CP/M command line when TXED was loaded.

You may specify one of the following device names in place of a disc name:

CON:	Keyboard or screen
TTY:	Keyboard or screen
LST:	Printer
LP:	Printer
RDR:	CP/M reader device
PUN:	CP/M punch device

If you do, the remainder of the file specification will be ignored and input or output will be directed to the specified device. This is useful only on rare occasions.

The file name consists of 1 to 8 characters, usually chosen to give an indication of the file's contents.

The file type consists of 1 to 3 characters, separated from the file name by a full stop. Certain file types are conventional on the 380Z computer, such as .TEX for text files and .BAK for backup files. Further examples will be found in Chapter 3.

 Copyright May 1980

COMMANDS THAT OPEN AND CLOSE FILES

The process of transferring text to a file from the text buffer is called writing to a file; the reverse process, reading, transfers text from a file to the text buffer. Before text can be written to, or read from a file, that file must be opened.

A file can only be opened for writing if no file of that name is present on the specified disc. The process of opening the file for writing enters the specified file name and type in the disc directory and makes the file ready to accept text (but does not transfer any text to the file). Text is written to the file by commands such as EX, N, W and QW. When writing is complete, the file must be closed. This is normally carried out automatically by the EX command but can also be performed when required by the EF command. Closing a file writes out the special character CTRL Z which marks the end of the file and updates the information about the file held in the disc directory.

A file can only be opened for reading if it already exists. The process of opening it makes the connection between the file and TXED, but again does not transfer any text. Text is read into the text buffer either by giving the R command, or by the implied R command that occurs in the EB or N commands. A file opened for reading does not need to be closed.

The commands in this group that include a file specification (shown as fsp) validate the specification before using it. An incorrectly constructed file specification can result in one of two errors. The message

```
?Bad dev?
```

means that an illegal device (disc) name has been found. The message

```
?Bad file spec?
```

means that the file name or type is at fault. In either case the command that contained the incorrect file specification is ignored.

Some of the other error conditions that can arise during file operations are included in the descriptions that follow. A complete list of error messages is contained in Appendix B.

EBfsp$ Edit a file

The EB command allows you to edit the file named by the file specification fsp and performs an automatic 'backup' operation on it.

An equivalent operation can be carried out by including fsp in the CP/M command line that loads TXED, viz.

```
TXED fsp
```

provided that the file already exists (see EW command if it does not).

The specified file is opened for reading and read into the text buffer. A temporary file is opened for output. When editing is complete, the EX command writes the contents of the text buffer to the output file and closes it. The original file is then renamed to type .BAK and the original file name and type transferred to the output file. The end result of the EB command is an edited version of the original file and a copy of its state prior to editing with the file type of .BAK (see Chapter 5).

If the file is larger than the capacity of the text buffer, the warning message:

More on input

is printed after the first portion of the file has been read in. When editing of that portion is complete, you may choose either to edit the second and subsequent portions of it by giving N commands, or to end the edit by giving the EX command.

If you change your mind in the middle of an edit and wish to abandon it, you can exit from TXED by typing the CTRL C command (see Chapter 8). This will return you to the Operating System with the state of the files as it was before the EB command was given. This is because the automatic renaming of files in an edit backup operation does not take place until the EX command is given. A temporary file with the file type of .$$$ will be present. It will be reused the next time you edit the file, or you can erase it in the normal way.

If the file specified by fsp cannot be found, the error message:

?File not found?

is printed and the command is aborted. The file types .$$$ and .BAK are required for the automatic backup process and if either is included in fsp, the error message:

?Illegal extension?

is printed and again, the command is aborted (extension is another name for file type).

EX End an editing session

The EX command completes the editing of a file and ends an editing session. The contents of the text buffer are written to the output file, followed by any unread text from the input file. The output file is then closed and control returns to the Operating System.

If no output file is open, the error message:

?No file?

 Copyright May 1980

is printed and the command is aborted.

The EX command is always safe. If the text buffer is empty, no output file is open, and no text remains to be read from the input file, the command merely returns to the Operating System. On the other hand, no matter how much text remains to be read from the input file, EX will copy it to the output file, then close the output file before exiting. Problems only arise when disc space becomes exhausted during the EX operation. Recovery from this error is discussed in Appendix B.

If the QX command has been issued prior to giving the EX command, the return to the Operating System is suppressed and TXED is restarted.

EI Initialise file system after changing a disc

The EI command resets the CP/M Disc Operating System and must be given after changing a disc. If you fail to do so and subsequently attempt to write to that disc, a 'Read Only' (R/O) error will be generated. Recovery from Read Only errors is described in Appendix B.

CAUTION: do not issue the EI command whilst a file is open for writing. The contents of that file will be lost.

ERfsp$ Open file for reading

The ER command opens the file specified by fsp and makes its contents available for reading by the R command.

If the file cannot be found on the specified disc, the error message:

?File not found?

is printed and the command is aborted.

When an EB command is active, an additional input file may not be opened until the whole of the original file has been read. This is to prevent inadvertent loss of data. If unread text remains in the original file, as indicated by the warning message 'More on input', an attempt to open another input file with an ER command will result in the error message:

?"EB" - more input?

and the ER command will be aborted. You should read in the rest of the original file with an R or N command before trying the ER command again.

 Copyright May 1980

EWfsp$ Open file for writing

The EW command opens the file specified by fsp for writing. If a file is already open for output, it is closed (see EF command) before the new file is created.

An equivalent operation occurs when fsp is included in the CP/M command line that loads TXED, viz.

TXED fsp

provided that the specified file does not yet exist. The message

New file

is displayed after TXED's title in this case (see EB command if file exists).

If the EW command was given and the file exists, the error message:

?File exists?

is printed and the command is aborted. If you are sure that you want to create a new file of that name, you will have to erase (QE) or rename (QR) the existing one before trying the EW command again.

If fsp was given on the CP/M command line and the file exists, an EB command is performed (see above).

When an EB command is active, a new output file may not be opened until the edit backup operation is complete. This is to prevent inadvertent loss of data. An attempt to open an output file with EW while an EB command is in progress will result in the error message:

?"EB" already active?

The EW command is aborted.

EF End file

The EF command closes the output file. If no file is open, the error message:

?No file?

is printed and the command is aborted.

CAUTION: If the EF command is issued while an EB command is in progress, the output file will be closed, and the automatic renaming of input and output files will take place, just as in an EX command. However the text in the text buffer will not

have been written to the output file, nor will any unread text from the input file. This is a dangerous situation to reach and should be avoided unless you know exactly what you are doing!

The EF command is not needed very often. Typical uses are when dividing up an input file into a number of output files, and after the QW command. Both these uses are illustrated in the Recipe Section, below.

COMMANDS THAT READ AND WRITE TO FILES

The four commands in this group transfer data between files and the text buffer.

R Read from input file

The R command reads text into the text buffer from the input file opened by an ER command. It can also be used to read additional text from a file specified an EB command or on the CP/M command line, if that file is larger than the text buffer. The numeric argument must be +1.

Text read in from the input file is always added to the end of the text within the text buffer regardless of the position of the text pointer, which is not changed. Up to three quarters of the space remaining in the text buffer is filled, to allow for the insertion and editing of additional text.

Text is read into the buffer until one of the following conditions occurs:

1. End of file is reached, denoted by reading a CTRL Z character. The CTRL Z character is not included in the text buffer.

2. Three quarters of the space remaining in the text buffer has been filled. In this case TXED continues reading text into the buffer until the end of a line is reached, unless this would overfill the buffer. The warning message:

   More on input

   is displayed.

3. There is room for fewer than 40 characters in the text buffer. The error message:

   ?No room?

   is printed.

4. A CTRL L (FORM FEED) character is read, signifying logical end of page. The CTRL L character is included in the text

 Copyright May 1980

buffer and is displayed as '↑L'.

After the 'More on input' message, the R command may be repeated, and, if there is only a small amount of text still to be read, may be successful in reading it all into the text buffer. With larger files, however, it is preferable to finish editing the first portion and then read in the second and subsequent portions with the N command (see below).

If no input file is open, the error message:

?No file?

is printed and the command is aborted. If there is no more text to be read from the the input file, the error message:

?End of file?

is printed and again, the command is aborted.

N Write to output file, read from input file

The N command writes the contents of the text buffer to the output file, clears the buffer, then reads more text from the input file. It allows the Next part of a large input file to be edited. The numeric argument must be +1.

The N command is functionally the same as the compound command B*WR and the same set of error conditions that can arise in the W and R commands apply.

The command is useful when editing a file that is larger than the capacity of the text buffer. When the first portion of such a file is read in to the text buffer, by an R or EB command, or by including the name of the file on the CP/M command line, the message:

More on input

will warn that more of the file remains to be read. When editing of the first portion is complete, the N command writes it to the output file and reads in the next portion for editing. This process is then repeated until the whole of the input file has been edited.

It should be noted that the EX command given at any stage of the editing of such a large file will always end the edit correctly.

nW Write n lines to output file and delete them

The W command writes n lines of text from the text buffer to the output file and then deletes them from the text buffer. A new text pointer position is calculated in the same way as for

the L and K commands (see Chapter 9). The numeric argument n may be positive or negative. The command then writes the text between the current text pointer and the calculated position to the output file. After the text has been written out it is deleted from the text buffer (as in the K command).

Deletion prevents accidental duplication of text in the output file. When the automatic deletion is undesirable,the S or QC commands can be used to make a copy of the selected portion of text before writing it to the file (see Chapter 9), or the text can be read back into the text buffer after writing by closing the output file with the EF command and opening it for input with ER.

If no output file is open, the error message:

?No file?

is printed and the command is aborted.

CAUTION: the W command is used mainly for splitting an input file into one or more smaller output files. You might use it, for example, to extract selected procedures from one program for use in another. The W command is not one that is needed regularly and you should make sure that you also understand the EW and EF commands before you use it. There is an example of the W command in the Recipe Section, below.

QW Output formatted text to file

The QW command moves the text pointer to the beginning of the text buffer, formats the text according to the formatting commands contained within it, and sends the formatted text to the output file.

An output file should have been opened by the EW command before the QW command is given, and must be closed with an EF command after it has finished. If no output file is open, the error message:

?No file?

is printed and the command is aborted.

Side effects of the QW command are that the text pointer is moved to the beginning of the text buffer and that the contents of the Macro/Save buffer are lost. Refer also to the descriptions of the QT and QP commands in Chapter 9, and to the Reference Section on formatting in Chapter 11.

There is an example of the QW command in the Recipe Section, below.

 Copyright May 1980

UTILITY COMMANDS

The three commands in this group are used for file maintenance.

QEfsp$ Erase file

The QE command erases the file specified by fsp. If the file cannot be found on the specified disc, the error message:

```
?File not found?
```

is printed and the command is aborted.

Refer to Appendix E if you wish to allow an ambiguous file specification in the QE command.

QRnewold Rename file

The QR renames the file named by the file specification 'old' to the file specification 'new'. For compatibility with CP/M, the command may also take the form:

```
QRnew=old$
```

If 'old' cannot be found on the specified disc, the error message:

```
?File not found?
```

is printed. If 'new' already exists, the error message:

```
?File exists?
```

is printed. In either case the command is aborted.

QLfsp$ List file directory on screen

The QL command lists the names of one or more files, as specified by fsp, from the directory of the specified disc unit.

If fsp is empty, as in the command:

```
QL$$
```

all the files on the logged in disc unit are listed. If a disc unit is specified, as in the command:

```
QLB:$$
```

the files on the specified disc are listed.

If the specified file cannot be found in the directory, the

message

```
Not found
```

is printed. The file specification may contain the ambiguous characters question mark (?), which matches any character, and asterisk (*), which matches any file name or file type. Thus the command:

```
QLB:*.*$$
```

is equivalent to the previous example and will list all the files on disc B, while the commands:

```
QL*.TEX$$
QLCHAPTER?.TEX$$
```

will list all files of type .TEX, and all files called CHAPTERn.TEX, where n is any letter or number, respectively.

The QL command disables the text window. It can be reenabled by the ED command (see Chapter 9).

COMMON FILE OPERATIONS

In the remainder of this Chapter we present some 'recipes' for the more common file operations. This section is not meant to cover all the possibilities.

STARTING A NEW FILE

The simplest way to start a new file is to include its file specification on the CP/M command line that loads TXED (see Chapter 3):

1.	TXED fsp	Load TXED, open file for writing.
2.	Type in and edit text	
3.	EX$$	Write text to output file, return to Operating System.

As an alternative, the file can be opened by the EW command after TXED has been loaded:

1.	TXED	Load TXED.
2.	Type in and edit text	
3.	EWfsp$$	Open file for writing.
4.	EX$$	Write text to output file,return

 Copyright May 1980

to Operating System.

EDITING AN EXISTING FILE

As with starting a new file, the simplest way to edit an existing file is to include its name on the CP/M command line that loads TXED:

1.	TXED fsp	Load TXED, open files.
2.	Add and edit new text	
3.	EX$$	Update and close files, return to Operating System.

As an alternative, the file can be opened by the EB command after TXED has been loaded:

1.	TXED	Load TXED.
2.	EBfsp$$	Open files for edit backup.
3.	Add and edit new text	
4.	EX$$	Update and close files, return to Operating System.

In both cases, the original file specification fsp will be transferred to the edited file and the file type of the old version will be changed to .BAK.

EDITING A LONG FILE

A 'long' file is one that contains more text than the text buffer can hold. It may be edited as follows:

1.	TXED fsp	Load TXED, open files.
2.	'More on input'	There is more input to read.
3.	Edit first portion	
4.	N$$	Write out current contents of text buffer, read in next part of file.
5.	Edit next portion	
6.	Repeat N$$ if necessary	

 Copyright May 1980

7.	EX$$	Update and close files, return to Operating System.

The same operation can of course be carried out starting with the EB command.

MAKING A COPY OF A FILE

The following procedure can be used to make a copy of a file, or to edit an old file, saving the edited version in a new file. Step 4 is optional:

1.	ERfsp1$$	Open fsp1 for reading.
2.	R$$	Read it in to the text buffer.
3.	EWfsp2$$	Open fsp2 for writing.
4.	QX$$	Suppress return to Operating System after EX.
5.	EX$$	Write text buffer to fsp2, close fsp2.

If necessary, editing of the file can be carried out between steps 2 and 3. It should be noted that the QX command could have been given in any of the previous examples.

MERGING TWO OR MORE FILES

Two or more files may be combined into a new file by reading them into the text buffer in sequence. The example generates a new file fsp3 from two old files, fsp1 and fsp2:

1.	ERfsp1$$	Open fsp1 for reading.
2.	R$$	Read it into the text buffer.
3.	ERfsp2$$	Open fsp2.
4.	R$$	Append it to fsp1.
5.	EWfsp3$$	Open fsp3 for writing.
6.	EX$$	Write text buffer to fsp3, close fsp3 and exit.

The procedure becomes slightly more complicated when the old files are too large to be held in the text buffer in their entirety. Here is one example of how to cope with this:

1.	ERfsp1$$	Open fsp1.
2.	R$$	Start reading it in.
3.	'More on input'	There is not enough room for the whole of fsp1.
4.	EWfsp3$$	Open fsp3 for writing.
5.	N$$	Write text buffer to fsp3, read in rest of fsp1.
6.	ERfsp2$$	Open fsp2.
7.	R$$	Append it to end of fsp1.
8.	EX$$	Write buffer and exit.

Note that the absence of the 'More on input' message after step 5 implied that all the rest of fsp1 had been read in.

SPLITTING A FILE INTO TWO

There are a number of ways of splitting up a file.

One way is to read it into the text buffer, delete everything except the first wanted section, then open a file for writing and exit. The process is then repeated for each of the other wanted sections.

An alternative is to read in the file as before, then write each wanted section to a separate file with the W command. We illustrate this second method:

1.	ERfsp1$R$$	Read in the file to be split.
2.	Move pointer to start of first wanted section	
3.	EWfsp2$$	Open fsp2 for 1st extract.
4.	nW$$	Write out extract, where n is its length in lines.
5.	EWfsp3	Close fsp2, open fsp3 for 2nd extract.
6.	Move pointer to start of second section to extract	
7.	nW$$	Write out 2nd extract.
8.	EF$$	Close fsp3.

 Copyright May 1980

LISTING A FILE ON THE PRINTER

A file can either be listed literally or after formatting. To list a file literally (including any formatting commands it contains):

1.	ERfsp$R$$	Read in file.
2.	B*P$$	Move pointer to beginning of buffer, print buffer contents.

Where the file is to be formatted according to the format commands contained within it, the procedure is:

1.	ERfsp$R$$	Read in file.
2.	QT$$	Format and print text.

The QP command would be used in place of QT if using single sheets of paper.

FORMATTING A LONG FILE

Where the file to be formatted is larger than the text buffer, it is necessary to insert the special formatting commands .BE and .EN at the beginning and end of the file to make sure that formatting information is carried over from one section to the next:

1.	ERfsp$R$$	Start reading in file.
2.	'More on input'	There is more input to read.
3.	Insert .BE command	
4.	QT$$	Format and print current contents of text buffer.
5.	*K$$	Clear buffer.
6.	R$$	Read in rest of file.
7.	Insert .EN command	
8.	QT$$	Format and print remaining text.

The example assumes that step 6 was successful in reading in the whole of the rest of the file. If not, go back to step 4.

 Copyright May 1980

Assuming that the .BE and .EN commands are in place, the compound command

```
ERfsp$*<RQT*K>$$
```

will format and print a long document automatically.

SAVING FORMATTED TEXT IN A FILE

When writing formatted text to a file with the QW command, a file must be opened with EW before giving the QW command, and closed with EF afterwards. The example assumes that the file to be formatted is already in the text buffer.

1.	EWfsp$$	Open file for writing.
2.	QW$$	Format text and write it to file.
3.	EF$$	Close file.

CHANGING A DISC DURING EDITING

If you wish to change a disc during editing, you must first make sure that any file open for writing is closed. The EI command should be given immediately after changing the disc:

1.	QXEX$$	Close file open for writing (if there is one).
2.	Change disc	
3.	EI$$	Reset Operating System.

Do not change a disc while a file is open for writing; a subsequent write command to that disc will corrupt the files on it.

 Copyright May 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

CHAPTER 11

FORMATTING REFERENCE SECTION

This Chapter contains the detailed descriptions of the text formatting commands available in TXED. Further information will be found in Chapter 7, which contains an introduction to text formatting, and in Appendix G, which describes the differences between TXED and TEX. The formatting commands are summarised in the Quick Reference Guide in Appendix A.

The Chapter starts off with some general definitions of the formatting process. Descriptions of the individual commands follow; these are presented approximately in the order that they would appear in a document and are often accompanied by general remarks on their usage with examples. Next, the operations involved in printing a long document and in outputting selected pages are discussed. The Chapter ends with a description of the special features which can be used with daisy wheel printers.

It is worth noting that this Manual has been prepared using TXED and that it therefore furnishes a major example of what can be achieved.

GENERAL DESCRIPTION OF FORMATTING

Text formatting is the process of neatly laying out a document on a suitable printer.

The most obvious function of a text formatter is to adjust each line of text to a predetermined length. In Chapter 7 we described how text which has been typed in to the computer without regard to line length can be processed in one of three ways:

1. It can be printed literally with preservation of the original line length, indentation and spacing. Literal mode is used mainly for printing lists or tables.

2. Lines can be filled, by packing as many words as possible from the text buffer onto an output line before printing it. This gives an appearance similar to typewriting, with an uneven right margin, hence its name of 'Ragged Right' mode.

3. As well being filled, lines can be justified to give an even right margin, by adding spaces between the words in the filled line before printing. This is called 'Fill and Justify' mode and, since it is the one most people want to use, it is selected as the initial state of the formatter.

Further functions of the text formatter include specifying the position of the body of the text on the page, beginning new lines, paragraphs and pages, controlling indentation, and setting up a running heading and folio which may include a page number etc.

FORMATTING COMMANDS

A formatting command comprises a dot (full stop), a two-letter command and perhaps some optional information. Each command must appear at the beginning of a line, with nothing on the line but the command and its arguments. For example:

```
.CE
```

centres the next line to be printed, and:

```
.SP 3
```

generates three line spaces (blank lines).

Where a numeric argument can follow a command, as in the previous example, it consists of a decimal number, possibly preceded by a plus (+) or minus (-) sign. An unsigned number is treated as an absolute value. For instance, the command

```
.IN 10
```

sets the current indentation level to column 10 (counting columns from zero). In contrast, a signed number is relative; the command:

```
.IN +10
```

increases the current indentation by 10. If this command followed the previous example, indentation to column 20 would now occur.

It is often more convenient to work with a relative value, particularly with the .TI command (see below).

The command letters can be in upper or lower case; for instance the two commands .CE and .ce have the same effect. The numeric argument can be separated from the command by one or more spaces. For example, the formatter considers the commands .SP3 and .SP 3 to be the same.

WHEN TEXT IS FILLED

In both Ragged Right and Fill and Justify modes lines of text are filled with words prior to printing. In Fill and Justify mode, they are also justified by adding spaces.

As each line is filled all extra spaces are removed, and in Ragged Right mode words end up separated by exactly one space. This has the advantage that additional spaces that have been typed accidentally at the end of a line or between two words do not appear in the output, but it does mean that special steps have to be taken when an extra space is required, for example, after a full stop. TXED allows you to define a special pad character which is printed as a space but will not be stripped off when a line is filled. For further details, refer to the descriptions of the .PC and .TI commands.

Completely blank lines at the start of a page are also ignored. Refer to

 Copyright May 1980

the section on leaving space for a diagram for a way of circumventing this.

BREAKS

Except when Literal mode is in effect, line filling proceeds continuously and if not interrupted, the text would be printed in one long paragraph. The commands which interrupt this process are said to cause a break. Any partially completed line is output, leaving the formatter ready to begin a new line, paragraph or page. Many commands implicitly cause a break and there is one, .BR, which does so explicitly, causing output to resume on the next line.

The list of formatting commands in the Quick Reference Guide in Appendix A identifies which commands do cause a break and which do not.

PRINTING A DOCUMENT

Formatting is carried out when a QT, QP, QV or QW command is issued. These commands are given in Command mode and are described in Chapters 9 and 10.

QT and QP output the formatted text to the printer, or to the screen if no printer option has been selected. Selection of a printer option is described in Appendix F. QT outputs the whole of the text without stopping and is intended for use with continuous or fan-fold paper. QP pauses at the end of each page to allow a new sheet of paper to be inserted in the printer. Type RETURN to resume printing. The QV command sends formatted output to the screen.

QW sends the formatted text to a file. The file must have already been opened with an EW command before giving the QW command and must be closed afterwards with an EF command. An example is given in the 'Recipe' section at the end of Chapter 10.

Immediately after a QT, QP, QV or QW command, the formatter sets the values of several of the formatting parameters such as line and page length. These initial values will be used until reset by formatting commands and result in a reasonable layout on the 9.5 by 12 inch (A4) perforated fan-fold paper commonly used with the Anadex DP-8000 and OKI Microline 80 printers. The initial settings are mentioned in the descriptions of the relevant commands and appear also in the Quick Reference list in Appendix A. If required, these settings can be changed by patching; refer to Appendix E.

Special steps are necessary when formatting a document that is too large for the text buffer to hold. The details are described at the end of this Chapter and the procedure is illustrated by a 'Recipe' at the end of Chapter 10.

After a minor revision it is often convenient to output only the corrected pages. Selective output of formatted text is achieved by the insertion of text marker characters and is described at the end of this Chapter.

 Copyright May 1980

COMMANDS THAT DEFINE PAGE LIMITS

The layout of text on the printed page depends upon three factors, the size of paper in use, the relationship of the paper to the printing mechanism, and the amount of space you want to leave between the text and the edges of the page.

The five commands in this group determine where the text is printed in relation to the page. There are commands to set the left and right margins, and the amount of space to be left blank at the top and bottom of each page. These commands are normally only given once, at the beginning of the text.

The margin settings depend upon the number of characters that can be printed on a line, which in turn depends on the number of characters printed per inch. This can vary from 5 to 16.5, but values of 10 or 12 characters per inch are common. Sometimes more than one setting will be available. Having decided on the number of characters per inch, determine the number of characters on a line by multiplying the characters per inch by the line length in inches. The size of the left and right margins should now be chosen. Subtracting the sum of these from the number of characters on a line gives the line length of the printed text. The left margin and (printed) line length are set by the .LM and .LL commands respectively, the right margin being whatever is left over.

The page length is set by the .PL command to the number of lines that can be printed on a page. Again printers vary, but line spacing of 6 lines per inch is common. Determine the page length setting by multiplying the number of lines per inch by the page length in inches. The .TS and .BS commands set the number of lines to be left blank at the top and bottom of the page respectively. Text will be printed on the remaining lines.

Assuming a paper size of 8.5 by 12 inches (approximately A4 size) and printer characteristics of 10 characters per inch and 6 lines per inch (typical settings on the OKI Microline 80, Anadex DP-8000 and Qume Sprint 5) suitable values for the above parameters are:

```
.LM 12     (left margin)
.LL 61     (line length)
.PL 72     (page length)
.TS 8      (top spaces)
.BS 8      (bottom spaces)
```

In practice, the exact positioning of the text will depend also on the mechanical relationship of the paper to the printing mechanism and small adjustments to these settings may be needed to achieve a symmetrical layout.

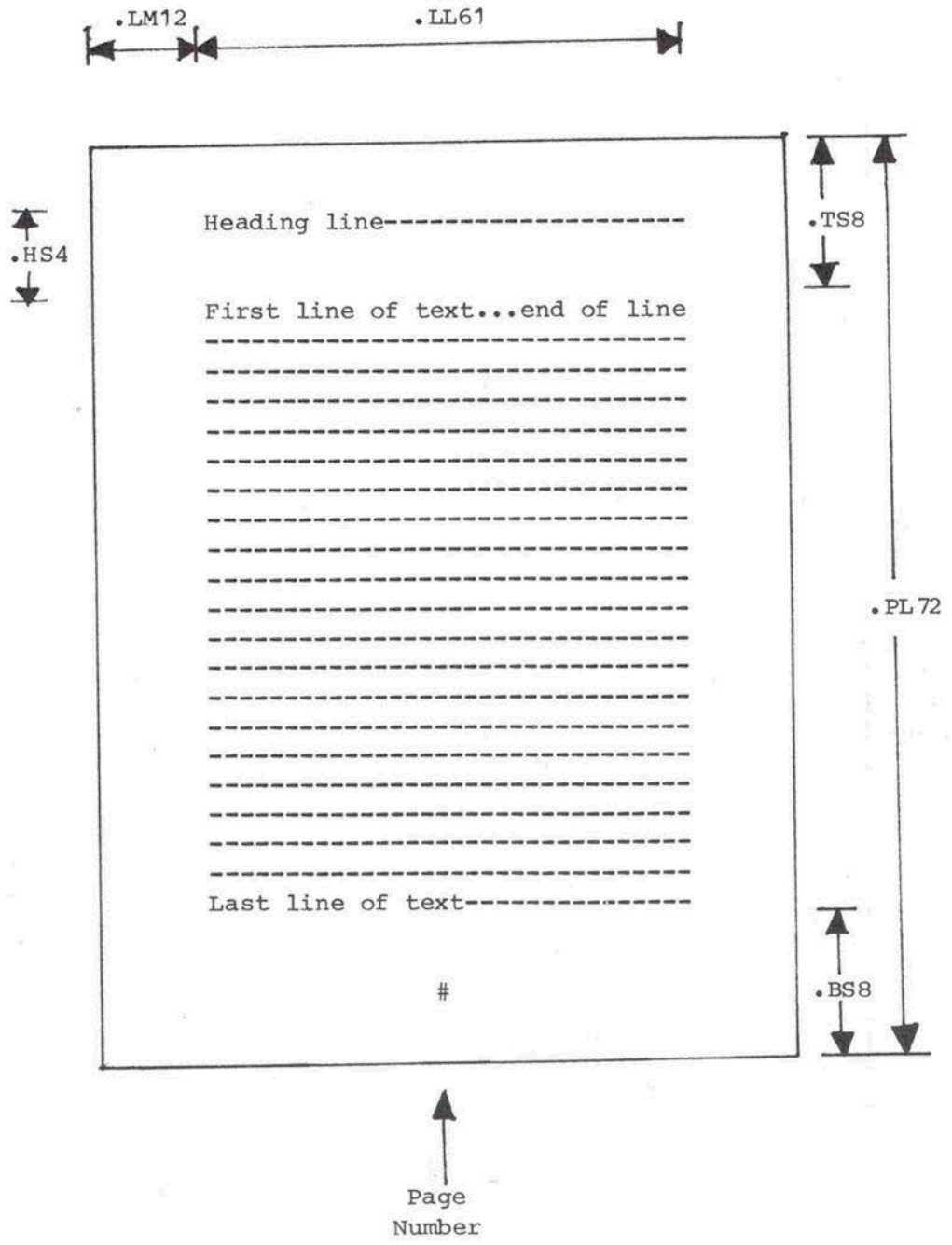


Fig 11.1 Showing the initial values assumed by the formatter in laying out a page. Note that the size of the right margin depends on the .LM and .LL settings and that the .TS and .BS commands set up the positions of the first and last lines in conjunction with the page length set by the .PL command.

 Copyright May 1980

.LMn Set left margin to n character positions

The .LM command sets the left margin. If the numeric argument n is absolute, that number of character positions is left blank at the beginning of each line. If n is relative, the current left margin setting is increased or decreased by n (it cannot be decreased below zero). The initial setting is 12 character positions. The left margin is set to zero during a QV command.

The left margin is usually set once at the beginning of a document and not changed thereafter. It applies to all text that is printed, including the running heading and folio. If it is changed, the right margin will change too as the .LM command does not affect line length.

To vary the left margin during the printing of a document, use the .IN command. This adjusts the line length to keep an even right margin.

.LLn Set line length to n character positions

The .LL command sets the line length of the formatted text. If the numeric argument n is absolute, the length of subsequent lines is set to that number of character positions. If n is relative, the current line length is increased or decreased by n (it cannot be decreased below zero). The initial setting is 61 characters.

The line length is usually set once at the beginning of a document and not changed thereafter. It applies to all text that is printed, including the running heading and folio. The only time that it is normally desirable to change the line length is when leaving space for a diagram on the right hand side of a page (see below).

The right margin is set by the .LM and .LL commands acting in combination.

.PLn Set page length to n lines

The .PL command defines the length in lines of a page of formatted text, and in combination with the .TS and .BS commands determines the number of formatted lines that will be printed on each page.

If the numeric argument n is absolute, the page length is set to that number of lines. If n is relative, the current page length is increased or decreased by n. The initial setting is 72 lines.

The page length is usually set once at the beginning of a document and not changed thereafter. It normally corresponds to the physical length of the paper being used.

 Copyright May 1980

TXED can be patched to output a special sequence of characters at the end of each formatted page (for example, the FORM FEED character). If your printer will skip to the top of a new page on receipt of such a control sequence, you could, if you wished to, work with a logical page length (set by the .PL command) that is less than the physical length of a page. Refer to Appendix E if you wish to implement this feature.

.TSn Set top space to n lines

The .TS command controls the number of lines to be left blank at the top of each formatted page.

If the numeric argument n is absolute, the top space is set to that number of lines. If n is relative, the current top space is increased or decreased by n. The initial setting is 8 lines.

The top space is usually set once at the beginning of a document and not changed thereafter. It may be blank or contain a running heading (see .MH command). If a heading is used, the top space should be set to at least two lines.

.BSn Set bottom space to n lines

The .BS command controls the number of lines to be left blank at the bottom of each formatted page.

If the numeric argument n is absolute, the bottom space is set to that number of lines. If n is relative, the current bottom space is increased or decreased by n. The initial setting is 8 lines.

The bottom space is usually set once at the beginning of a document and not changed thereafter. If automatic page numbering is used, the page number will be printed within the bottom space (see below). The bottom space may be otherwise blank, or contain a running folio (see .MF command). If page numbering and/or a folio is used, the bottom space should be set to at least two lines.

PAGE NUMBERING (PART 1)

Automatic numbering of pages will occur by default, starting from 1, and unless a folio has been defined, the page number will be centred horizontally and vertically within the bottom space.

The two commands in this group control whether or not page numbers are to be printed and can alter the value of the page number. The .MH and .MF commands determine where the page number is to be printed (see next two sections).

 Copyright May 1980

.OP Omit page numbers (and folio)

The .OP command suppresses the automatic printing of page numbers at the foot of each page. If a folio has been defined by the .MF command, it is also suppressed.

Page numbering can be resumed by issuing a .PN or .BPn command.

.PNn Enable page numbering, set page number

The .PN command turns on the automatic numbering of pages and sets the current page number.

If n is absolute, the number of the current page is set to n. If n is omitted, a value of 1 is assumed. If n is relative, the number of the current page is increased or decreased by n.

Automatic page numbering can also be restored by the .BPn command (see below) which is similar to the .PNn command but also begins a new page.

It is often convenient to keep the chapters of a book in separate files and to print them separately. By including a .PNn command at the beginning of each chapter, the pages of the book can still be numbered in sequence. Thus if chapter 2 ends on page 41 and the command .PN 42 is included at the start of chapter 3, the first page of chapter 3 will be numbered page 42.

You may wish to suppress the page number on the title page of a document but start numbering the pages from page 2. An .OP command included at the beginning of the document, prior to the end of the first page, will suppress the page number at the foot of that page. A .PN 2 command embedded somewhere within the text of the second page will restore page numbering and the second page will be numbered page 2.

RUNNING HEADING AND FOLIO

The commands in this group control the appearance and position of the running heading and folio (footing) which can appear at the top and bottom of each page in the top and bottom spaces.

.MH//// Define running heading

The .MH command defines a running heading that will be printed at the top of each page in the top space. Its position in the top space is set by the .HS command. The initial setting is for there to be no running heading.

Up to three sections of the heading can be defined which will be left justified, centred, and right justified, respectively. The first non-blank character following the command is taken as a delimiter separating the text that follows into three fields

and can be any character that does not occur in the heading. For example, the .MH command for this Chapter of the manual is:

```
.MH/TXED Text Editor//Formatting Reference/
```

in which the slash character (/) is the delimiter. The strings 'TXED Text Editor' and 'Formatting Reference' are defined as the left and right running headings, and the centred heading is blank. On the other hand, the command:

```
.MH**Operator's Manual**
```

would result in a centred heading of 'Operator's Manual' and blank left and right headings. In this case an asterisk (*) has been used as the delimiter.

The running heading can be changed from time to time by including a series of .MH commands in the text. The three subheadings are laid out in accordance with the current settings of the left margin and line length and the new heading will appear at the top of the next page to be printed.

If any characters with special significance (for example, the pad, bold and underline characters, to be described later) occur within the heading strings, they are processed according to their definitions at the time the .MH command is encountered.

.HSn Set heading space to n lines

The .HS command controls the distance in lines between the running heading and the first line of the text.

If the numeric argument n is absolute, the heading space is set to that number of lines. If n is relative, the current heading space is increased or decreased by n. The initial setting is 4 lines.

The heading space is usually set once at the beginning of a document, immediately after setting the top space. It is counted upwards from the first line of the text and includes the heading itself. Thus with the default heading space of 4 lines there are three blank lines between the heading and the first line of the text.

The top space should be large enough to include the heading space.

.MF//// Define folio

The .MF command defines a folio or running footing that will be printed at the bottom of each page in the bottom space. Its position in the bottom space is set by the .FS command. The initial setting is for the folio to consist only of a page

 Copyright May 1980

number (see next section).

Left, centre and right sections of the folio can be defined in the same way as for the .MH command (see above). The folio for this Chapter is defined by the command:

```
.mf/Research Machines/11.#/Copyright May 1980/
```

The significance of the hash character (#), which defines the position of the page number, is explained in the next section. The folio can be changed from time to time in the same way that the heading can, and characters with special significance are processed in the same way.

Note that folio will not be printed when page numbering is turned off by the .OP command.

.FSn Set folio space to n lines

The .FS command controls the distance in lines between the running folio and the last line of the text.

If the numeric argument n is absolute, the folio space is set to that number of lines. If n is relative, the current folio space is increased or decreased by n.

The initial setting is zero. A setting of zero forces the folio to be centred vertically within the bottom space as set by the .BS command.

The folio space is usually set once at the beginning of a document, immediately after setting the bottom space. It is counted downwards from the last line of the text and includes the folio itself. Thus with a folio space of 4 lines there will be three blank lines between the last line of the text and the folio.

The bottom space should be large enough to include the folio space.

PAGE NUMBERING (PART 2)

As mentioned earlier, the .MH and .MF commands can be used to define the position in which the page number will be printed. In order to do this, a special symbol, the 'number character', must be defined:

.NCc Define page number character

The .NC command defines the character c which follows to be the 'number character'. c can be any character. The initial setting is the hash character. If the character argument is omitted the special significance of the previously selected character is turned off.

The number character has special significance only within a heading or folio defined by the .MH or .MF commands. When the heading or folio is printed, any occurrence of the number character will be replaced by the current page number.

It can now be seen that the initial setting of the folio corresponds to the pair of commands:

```
.NC#
.MF//#//
```

which results in the page number being printed in the middle of an otherwise blank folio line.

This technique allows great flexibility. The folio of the current Chapter (given as an example in the description of the .MF command) illustrates how a Chapter number can be added to a page number. Another possibility is to define a folio such as:

```
.MF//Page # of 9//
```

which will number the pages in the form 'Page 1 of 9', etc.

The page number is usually printed at the bottom of the page, but it can equally well be located in the heading if required. Note that the .OP command will not suppress the page number in this position, since it does not affect the heading.

COMMANDS THAT AFFECT THE TEXT LAY OUT

The first three commands in this group determine whether the printed text should be both filled and justified, filled only, or output literally. Commands are also available to set the vertical line spacing. By means of these, each new line in the text buffer can be made to correspond to one, two or more line spaces on the printer.

.FJ Fill and justify

The .FJ command adjusts the text by adding or taking away words until the line length is as near as possible, but no greater than, the length set by the .LL command, and then if necessary by adding spaces to reach that length exactly.

Fill and Justify mode is the initial state of the formatter, and it is in this mode that this Manual (and this paragraph) has been printed.

.RR Fill to ragged right margin

The .RR command adjusts the text by adding or taking away words until the line length is as near as possible, but no greater than, the length set by the .LL command. In contrast to the

 Copyright May 1980

.FJ command, no additional spaces are added and as a result the right margin is uneven.

This description of Ragged Right mode has been printed with the .RR command in effect. The appearance resembles typewriting and many people find they prefer this mode, particularly for correspondence. Although the overall appearance is not so neat, it may be felt that words are more easily read and understood if the spacing between them remains the same. The use of automatic justification may also lead to gullys of spaces on consecutive lines. After this short example, we return to Fill and Justify mode, leaving the reader to make his or her own choice.

.LI Literal mode

The .LI command discontinues the filling and justification functions of the formatter and provides a method of printing text in the format in which it was typed.

For example, if the text buffer contains:

```
.LI
Alpha
Bravo
Charlie
Delta
.FJ
```

the list will be printed:

```
Alpha
Bravo
Charlie
Delta
```

rather than:

```
Alpha Bravo Charlie Delta
```

which is the case when the .LI command is not present.

Literal mode is cancelled by any other formatting command, the formatter reverting to either Fill and Justify or Ragged Right mode, whichever was in effect most recently. It is usual to terminate Literal mode with a layout command which is already in force, such as .FJ or .SS.

Within Literal mode, blank lines and lines beginning with a TAB character have no special meaning. This is in contrast to .FJ and .RR modes where blank lines and lines beginning with a TAB character are implicit commands (.SP and .PP).

Whilst in Literal mode, the margins, heading and folio, vertical line spacing, and any indent are maintained. TAB

characters are expanded according to the current setting of the ET command (see Chapter 9), starting from the first printing column, that is, from the column specified by the sum of the current .LM and .IN settings. The result of this is that tables which have been laid out using TAB characters will have a similar appearance on the printed page as they have on the screen. Long lines are not truncated at the right margin, however, and will be limited in length only by the mechanical constraints of the printer.

.SS Set single line spacing

The .SS command sets the vertical line spacing to single spacing. This is the initial condition.

.DS Set double line spacing

The .DS command sets the vertical line spacing to double spacing.

.LSn Set multiple line spacing

The .LS command sets the vertical spacing between lines to n line spaces. n must be absolute.

The .SS, .DS and .LS commands have a similar effect to the line space lever on a typewriter. Single line spacing, set by .SS or .LS 1, is the initial condition. Double line spacing, set by .DS or .LS 2, leaves room for corrections between the lines and is useful for draft reports. Multiple spacing, set by the .LS command with n greater than two, can be useful on special occasions.

STARTING A NEW LINE

As explained earlier, many commands implicitly cause a break in line filling, forcing out any partially completed line, before carrying out their specific function. The three commands in this group do so either explicitly, or implicitly prior to starting a new line or a new paragraph.

.BR Break line filling

The .BR command causes a break in the filling of the current line and causes it to be printed. Subsequent text will begin on a new line. If a .BR command immediately follows another command that has caused a break, nothing will happen.

A typical use of the .BR command is in printing a list. Referring back to the example used in the .LI command, an alternative method of printing the list is:

```
Alpha
.BR
Bravo
.BR
Charlie
.BR
Delta
.BR
```

The result is similar, namely:

```
Alpha
Bravo
Charlie
Delta
```

As noted at the beginning of this Chapter, those commands which cause a break implicitly are identified in the Quick Reference Guide in Appendix A.

.SPn Output n line spaces

The .SP command outputs one or more blank lines, the number depending both on the value of the numeric argument n and on the line spacing currently in force, as set by the .SS, .DS and .LS commands. Line spacing is initially set to 1 (.SS).

If n is omitted, a value of one line space is assumed. Otherwise n line spaces are output, each line space comprising 1, 2 or more blank lines. n must be absolute.

Except in Literal mode, a blank line in the text is equivalent to an .SP 1 command. Thus, the commands:

```
This is line 1
.SP 1
This is line 2
```

and:

```
This is line 1

This is line 2
```

are equivalent, and both will be output as:

```
This is line 1

This is line 2
```

The .SP command is therefore used mainly when more than one line space is wanted.

It is important to note that the .SPn command will not space past the end of a page. This is because completely blank lines

at the beginning of a page are ignored. This is discussed further in the section on leaving space for diagrams, below.

.PPn Start new paragraph

The .PP command starts a new paragraph by outputting one blank line space and indenting the next line of text n character positions. n must be absolute.

The blank line space will consist of 1, 2, or more blank lines, as set by the .SS, .DS, or .LS commands (the initial setting being 1).

If the numeric argument n is omitted, the last specified value of n will be used. The initial setting is 6 character positions.

We have not used the .PP command in the formatting of this Manual, preferring the 'blocked' form of layout. An example of the .PP command is:

```
Dear Sir,
.PP
With reference to yours of the 20th
ult., I am glad to be able to report...
```

which will result in:

```
Dear Sir,

     With reference to yours of the 20th ult., I am glad
to be able to report...
```

Except in Literal mode, a line beginning with a TAB character is treated as though the TAB character is a .PP command. A blank line space is output and the remainder of the line indented by the current .PP setting.

The .PPn command is equivalent to the command .SP 1, followed by .TIn (see below).

 Copyright May 1980

STARTING A NEW PAGE

Three commands can force the formatter to begin new page:

.BPn Begin a new page

The .BP command causes a new page to be started.

If the numeric argument n is omitted, the command simply begins a new page. The new page number will be one greater than the preceding one. If the page number and the folio have been suppressed by the .OP command, they remain suppressed.

If n is absolute, the command begins a new page and numbers it page n. If n is relative, a new page is begun with the page number increased or decreased by n. In both these cases, printing of the page number and folio will be resumed if it has been suppressed by an .OP command.

The .BP command is really a dual purpose command. Used without a numeric argument, it simply begins a new page. With a numeric argument, its main usefulness is in leaving a gap of a page or two in the text, for example for a diagram. The page containing the diagram can be prepared separately and numbered using the .PN command.

.CPn Conditionally begin a new page

The .CP command begins a new page if, and only if, the next n lines of text will not fit on the current page. The numeric argument n must be absolute.

A typical application of the .CP command is to ensure that an item of text such as a table does not become split across the bottom of a page. A .CP command placed at the beginning of such an item declares that n lines will be needed for it; a new page is started if less than n lines remain. The .CP command is also useful for preventing section titles appearing in isolation at the bottom of a page.

.PAn Advance n pages

The .PA command causes n blank pages to be output. If the numeric argument n is omitted a value of 1 is assumed. n must be absolute.

The pages will contain a heading, a folio, or a page number if the appropriate options are in effect, or they may be completely blank.

Possible uses of the .PA command are to force the beginning of a Chapter onto an odd numbered page, or to leave room for comments to be added to a draft text.

 Copyright May 1980

The .BP and .PA commands look fairly similar and in fact, when 'n' is omitted, have the same effect. The main difference between them is that the .BPn command advances to page n without outputting the intervening pages whereas the .PAn command outputs n blank pages.

INDENTATION

Two commands are available to vary the distance between the left margin set by the .LM command and the left hand edge of the text. This distance is called the indent. When an indent is in effect, the line length set by the .LL command is automatically reduced, such that the right margin stays constant. The current indent can be increased, decreased, or cancelled. It cannot be reduced to less than zero, i.e. you are never allowed to print to the left of the left margin.

.INn Indent subsequent text

The .IN command causes the text on subsequent lines to be indented by the specified number of character spaces, until either the indent is cancelled or another indent command is given. The initial setting is zero (no indent).

If the numeric argument n is absolute the indent is set to that number of character spaces. If n is relative the current indent is increased or decreased by n (it cannot be decreased below zero).

It is important to distinguish between the .LM (left margin) and .IN (indent) commands since both can change the apparent left margin. The .LM command is intended for setting the left hand edge of the text (including heading and folio) in relation to the paper. The .IN command is used to move the left hand edge of the text to the right from time to time, mainly to improve readability. Heading and folio are not affected by the .IN command.

.TIn Temporary indent

The .TI command causes a temporary indent of the specified number of character spaces, which applies for one line only.

If the numeric argument n is absolute a temporary indent of that number of character spaces from the left margin occurs. This is independent of any indent currently set by the .IN command, and can be greater or less than it.

If n is relative the current indent is increased or decreased by n for one line.

 Copyright May 1980

.CI Cancel indent

The .CI command cancels an indent set by a previous .IN command. Since this is equivalent to an indent of zero, the command .IN 0 has the same effect.

The .IN and .TI commands have been used freely throughout this Manual with the aim of improving its legibility.

The convention used for presenting examples is to print them on a separate line, indented by 10 spaces from the main body of the text. As an example of an example, the commands:

```
.IN 10
.LI

This is an example

.CI
```

result in:

This is an example

Note that while the indented example (which may consist of more than one line) is output in Literal mode, Fill and Justify mode is resumed on encountering the .CI command.

A further useful technique, involving both the .IN and .TI commands, is illustrated by the following extract from the description of the R command (Chapter 10):

'Text is read into the buffer until one of the following conditions occurs:

1. End of file is reached, denoted by reading a CTRL Z character. The CTRL Z character is not included in the text buffer.

2. Three quarters of the space remaining in the text buffer has been filled. In this case TXED continues reading text into the buffer...'

The point illustrated here is the hanging indentation of the numbers '1.' and '2.' to the left of the paragraphs they number. For a full understanding of how this is achieved, the 'pad character' must also be mentioned. This is a special character which is printed as a space but treated as part of a word when the text is formatted. When two words are separated only by one or more pad characters, additional space characters will not be inserted between the words during justification. The .PC command (see below) is used to define a pad character; any character can be nominated but normally a punctuation character is chosen.

The extract from the description of the R command starts with an indent of 10 in effect. Immediately before the first numbered paragraph, the

commands are:

```
.IN +4
.TI -4
1.&&End of file is reached...
```

the pad character having been defined to be ampersand (&). The .IN +4 command causes the indent to be increased by 4, from 10 to 14, but for the first line only, the .TI -4 command reduces it back to 10. This results in the number and the rest of the first line of paragraph 1 being printed four columns to the left of the remainder of the paragraph. The .TI -4 command is repeated at the beginning of each of the remaining paragraphs.

But why use pad characters? The problem is that Fill and Justify mode is in effect. If we had just typed a space between '1.' and 'End', the number of spaces that would be printed between these two words would be indeterminate, depending upon the amount of justification required. There would be no guarantee that the word 'End' would be lined up vertically with the remaining lines of the paragraph which are indented by 14 spaces. However, because we have used pad characters, the formatter considers the first word to be:

```
1.&&End
```

No spaces can be inserted before 'End' and the letter 'E' will occur exactly 4 character positions after the '1'.

CENTERING AND FULL INDENTATION

.CEn Centre n lines of text

The .CE command centres the next n lines of text. If the numeric argument n is omitted, a value of 1 is assumed. n must be absolute.

The .CE command is used to centre chapter and section headings. For example:

```
.CE 2
SECTION FIVE
RELIABILITY AND MAINTENANCE
```

results in:

SECTION FIVE
RELIABILITY AND MAINTENANCE

Note that the current indent is ignored when centering.

The centering action is terminated by any other formatting command (including the implicit .SP 1 command caused by a completely blank line) even if n lines have not yet been printed. This means that if you don't want to count lines, you

can put:

```
.CE 99
lots of lines
&
to be centred
.BR
```

which will centre an arbitrary number of lines. Note the use of a line consisting only of a pad character (&). This trick allows blank lines to be included amongst a group of lines to be centred. As mentioned above, a completely blank line, or one consisting only of space characters, would terminate the centering command.

No filling is done on centred lines.

.IFn Indent n lines fully

The .IF command causes the next n lines of text to be indented fully such that the last character on each line is next to the right margin. If n is omitted, 1 is assumed. n must be absolute.

The .IF command is useful for printing addresses and similar items. For example:

```
.IF 4
Research Machines Ltd.
P.O. Box 75
Mill Street
OXFORD OX2 0BW
```

results in:

Research Machines Ltd.
P.O. Box 75
Mill Street
OXFORD OX2 0BW

Like the .CE command, the .IF command is cancelled by any other command, and fully indented lines are not filled.

MISCELLANEOUS COMMANDS

.PCc Define pad character

The .PC command defines the character c which follows to be the 'pad character'. c can be any printing character whose ASCII code is greater than 20 hex, but usually a punctuation character is chosen. If the character argument is omitted the special significance of the previously selected character is turned off.

The pad character is printed as a space but is treated as part of a word when the text is formatted. If two words are separated only by pad characters, they will not be spit onto separate lines when line filling occurs, and additional spaces will not be inserted between them during justification.

The pad character has a number of applications. Use in constructing a hanging indent has already been mentioned (see .TI command). A pad character can be used to keep combination words together. In this Chapter, we have typed examples of formatting commands in the form:

```
.TI&-4
```

where the ampersand (&) is defined as the pad character, in order to ensure that '.TI' and '-4' stay together.

Pad characters can also be used to ensure that full stops that end sentences are followed by at least two spaces. Recalling that TXED reduces multiple spaces into a single space before filling and justifying, it will be evident that it makes no difference how many ordinary spaces are typed after a full stop. But if a full stop at the end of a sentence is always followed by a pad character and then by a space, a minimum of two spaces will be printed and perhaps more if the line is justified. Two sentences earlier in this paragraph, the text buffer reads:

```
after a full stop.& But if a full stop...
```

To prevent an anomaly where such a 'padded' full stop occurs at the end of a line after filling, pad characters falling at the end of lines after filling are removed before justification.

The pad character is normally defined as an otherwise unwanted punctuation character at the beginning of the text buffer and not changed thereafter.

.IG Ignore comments

The .IG command causes the formatter to ignore the text which follows, until another formatting command is encountered. It allows the insertion of comments into the text buffer such as:

```
.IG
Chapter 3 (disc version)
Revised 15 May 1980
.ST
```

The .IG command does not cause a 'break'.

 Copyright May 1980

.ST Stop previous command

The .ST command stops whatever special operation the previous command is doing, but without causing a 'break'. It is occasionally useful to terminate a command which itself does not cause a break, such as the .IG command.

.CCc Insert control character

The .CC command converts the character c which follows into a control character and outputs it to the printer. c should be either an upper case letter in the range A-Z or one of the punctuation characters with ASCII codes 5B-5F inclusive (refer to a table of the ASCII character set).

The .CC command is useful for enabling special printer options. For example, on the Anadex DP-8000 printer, the command .CCN causes the next line to be printed in double width characters. Apart from the special daisy wheel printer options (see below), the .CC command is the only way of outputting control characters directly to a printer since control characters that are present literally in the text buffer are ignored.

Since the .CC command causes a break, the control character will be output before the first character on a line. It does not count towards the length of that line however.

LEAVING SPACE FOR A DIAGRAM

You may wish to reserve space on a page for a diagram. Space can be reserved on the left, on the right, at the top, or at the bottom of a page.

Space is reserved between the left margin and the left hand edge of the text by setting an indent with the .IN command. This remains in effect until cancelled.

To leave space on the right you must temporarily reduce the line length setting with the .LL command. It is often convenient to use relative values of n for this, for example:

```
.LL -20
lines of text
to the left
of a diagram...
.LL +20
```

This command sequence would leave a gap of twenty character positions between the right hand edge of the text and the 'normal' right margin.

To reserve space at the top of page, starting where the text would normally begin, requires a trick. As mentioned previously in the description of the .SP command, completely blank lines and lines

consisting only of space characters are ignored at the beginning of a page. The trick is to start a page with a line consisting only of a pad character (see .PC command). After this special line, the .SP command can be used in the normal way. Thus, if you wished, say, to leave 10 blank line spaces at the beginning of a page, an appropriate command sequence would be:

```
.PC&
.BP
&
.SP 9
beginning of text
```

whereas the sequence:

```
.BP
.SP 10
beginning of text
```

would leave no gap, the text on the new page starting at its normal position.

Space is left at the bottom of the page by simply starting a new page early with the .BP or .PA command.

FORMATTING LONG FILES

A long file is one that is larger than the text buffer can hold. When you attempt to edit such a file, the warning message:

```
More on input
```

is printed at the bottom of the screen after the first portion has been read into the text buffer. Since the formatter works on the contents text buffer, it is necessary to format such a long file in stages.

Normally, each time a QT, QP or QW command is issued, formatting parameters such as line and page length are set to their initial values and the formatter starts off in Fill and Justify mode. The .BE command command signals to the formatter that it should not reset these values on a subsequent QT, QP or QW command. This permits you to read in the second and subsequent portions of a long file and format them as continuations of the first portion. The .EN command flushes out the last line of text from the last filling of the text buffer and causes the formatting parameters again to be initialised each time a QT, QP or QW command is given.

.BE Begin formatting a long file

The .BE command signals to the formatter that this is the start of a piece of text that is longer than the current contents of the text buffer. Until a .EN command is encountered, you may issue the QT, QP or QW commands as often as necessary without causing the formatting parameters to be reset to their initial

values. In addition, the last line of the text may not be printed if there is a possibility that it can be continued using words from the next filling of the text buffer.

The .BE command should appear on the first line of the file and will set the formatting parameters to their initial values. In addition, the last character in each filling of the text buffer should be a RETURN character. If the text is read in in the normal way, with an EB, R or N command, TXED will arrange this automatically (see R command, Chapter 10).

.EN End formatting of a long file

The .EN command terminates the formatting of a long file. It should appear on the last line of the file, after the end of the text. The .EN command ensures that the last line of text will be printed and cancels the .BE command. Subsequent QT, QP or QW commands will again set the formatting parameters to their initial states.

Refer to the 'Recipe' section at the end of Chapter 10 for an example of the .BE and .EN commands.

SELECTIVE OUTPUT

After revising a document it is often desirable to reprint only the pages that have been changed. Selective output is achieved by the insertion of text marker characters.

The insertion of text markers into the text buffer is described in Chapters 4 and 10. During formatting a text marker acts as switch, turning the formatted output successively off and on, the initial state being on. A change in state can only take place at the beginning of a page; a text marker must be therefore be inserted within the text of the page before the one with which printing should start or stop.

Suppose you have revised a document and wish to reprint only pages 3 to 5. A text marker is inserted at the beginning of the text buffer so that output will initially be switched off. A second marker must be inserted somewhere within the text of page 2. This will switch on the formatted output at the beginning of page 3. Similarly, a third marker within the text of page 5 will turn off output again at the end of that page.

This process can be repeated as often as necessary. Thus further markers within the text of pages 19 and 20 would result in page 20 being printed also.

In the case of long documents where the .BE and .EN commands are in use, the first text marker should be placed on the line after the .BE command as this command causes all initial values to be restored including the selective output switch which is turned on.

Remember that text markers are removed when the contents of the text

buffer is written to a file.

SPECIAL FEATURES FOR THE QUME SPRINT 5

The formatting commands described in this section apply in general only to the Qume Sprint 5. These commands define special characters that allow underlining, bold face printing, superscripts and subscripts, and access to the two 'hidden' characters on the daisy wheel (one of which is often the U.K. pound sign). Bold face and underlining can be used with an 80 character board with the QV command.

To allow documents that will eventually be printed on a daisy wheel printer to be developed on an ordinary matrix printer, the .QN command instructs the formatter to ignore all occurrences of these special control characters.

The commands are:

.UCc	Underline
.BCc	Bold face
.E1c	Escape
.E2c	Superscript
.E3c	Subscript
.E4c	Backspace before printing
.E5c	Don't advance after printing

Like the .PC and .NC commands, these commands designate the character c that follows to have a special function. Again, like the .PC command, the special significance of the character is cancelled by giving the command without a character after it.

.QN Ignore special characters

The .QN command is inserted at the beginning of the text buffer (after a .BE command if present) in order to suppress the special significance of characters defined by the .UC, .BC, and .E1 to .E5 commands. It allows text that is intended to be printed finally on a daisy wheel printer to be developed on an ordinary matrix printer that does not support such facilities.

.QY Process special characters

The .QY command restores the significance of the special daisy wheel characters, reversing the effect of the .QN command.

.UCc Define underline character

The .UC command designates the character c which follows to be the underline character. c may be any printing character. If c is omitted, the special significance of the previously selected character is lost.

 Copyright May 1980

Each subsequent appearance of c reverses the underlining 'switch', turning it on if it is off, or vice versa. For example:

```
.UC&
&Underlined words and more words&
.UC
```

temporarily defines ampersand (&) to be the underline character and causes the phrase:

Underlined words and more words

to be underlined. Underlining is confined to individual words if the words are separated by space characters; if the words are separated by pad characters, the spaces between them are underlined too. Thus:

```
.UC&
.PC!
&FORMATTING!REFERENCE!SECTION&
.UC
```

will be printed as:

FORMATTING REFERENCE SECTION

just as it is at the beginning of this Chapter.

.BCc Define bold face character

The .BC command designates the character c which follows to be the bold face character. c may be any printing character. If c is omitted, the special significance of the previously selected character is lost.

Each subsequent appearance of c reverses the bold face 'switch', turning it on if it is off, or vice versa. For example:

```
.BC&
&This is a bold face sentence.&
.BC
```

temporarily defines ampersand (&) to be the bold face character and causes the phrase:

This is a bold face sentence.

to be printed in bold face.

Bold face is achieved on the Qume Sprint 5 by overprinting the characters after stepping the carriage 1/120 inch to the right, and on an 80 character board by dimming. It is dramatic but not always entirely accurate. Bold face and underlining may be

mixed.

.E2c Define superscript character
.E3c Define subscript character

The .E2 and .E3 commands designate characters which cause the following text to appear as a superscript or subscript, respectively. There must be as many superscript as subscript commands. For example, commands of the form:

```
.E2&
.E3!
C!2&H!5&OH is ethanol.    2&3! = 8.
```

will print

C_2H_5OH is ethanol. $2^3 = 8$.

As with the .UC and .BC commands, the special significance of the characters designated by the .E2 and .E3 commands is lost when no character argument is supplied.

.E1c Define ESC character
.E4c Define BACKSPACE character
.E5c Define overprint character

Three more special characters, designated by the .E1, .E4 and .E5 commands allow the ESC character, the BACKSPACE character, and the character sequence ESC N, which suppresses carriage movement for one character, to be output.

The .E1 command allows the two extra characters on a daisy wheel to be printed. These are accessed by sending the printer the sequences ESC SPACE and ESC / respectively. On many daisy wheels, the U.K pound sign is printed by sending ESC SPACE. With such wheels you might type:

```
.E1&
.PC!
The price is &!99.95 per gallon.
```

thus printing the pound sign next to the first 9. The pad character between the '.E1' character and the 9, rather than a space character, ensures that no additional spaces will be added during line filling. (The pound sign is not available on the daisy wheel we are using to print this Manual so we can't print this example.)

The .E1 character can also be used to perform special functions on the OKI Microline 80 printer. See Appendix F.

The .E4 and .E5 commands allow overprinting. For example:

```
.E4&
The number is 0&/0&/0&/1
```

prints:

The number is ØØØ1

or alternatively:

```
.E5&
The next number is &0/&0/&0/2
```

which prints:

The next number is ØØØ2

amounting to the same thing.

All the commands in this section are set up initially for the Qume Sprint 5 daisy wheel printer. A restriction is that the characters printed as a result of the .E1 to .E5 commands cannot be underlined nor can they be in bold face. Each command causes TXED to output a specific string of characters. Refer to Appendix E for details of how to alter these strings if you wish to change them to suit other printers.

CHAPTER 12

MACRO COMMANDS

A Macro Command is a term used in computer programming to refer to a single 'large' command which can perform the work of a series of commands (from the Greek word makros, meaning large or long).

TXED allows you to define a macro command in terms of one or more Command mode commands and to execute it by a single command from Immediate or Command mode. There is provision only for the definition of a single macro but the definition can be changed as often as necessary.

We have had to defer the description of the macro command to this late stage in the Manual because its understanding requires a good knowledge of Command mode. Even so, the macro command is one of TXED's most useful facilities.

Uses of macros range from simply extending the range of Immediate mode commands (e.g. defining a command to move the pointer to the end of the line) to making a two column listing or printing a standard letter. Both types of use are covered in this Chapter. It will be helpful to read this Chapter in conjunction with Chapter 6 which describes some fairly similar operations carried out in a different way.

DEFINING AND EXECUTING A MACRO

The easiest way to execute a macro is simply to type the Immediate mode command CTRL X:

CTRL X Execute macro command.

But before you can do this you must define the macro command that will be executed, by using the M command (in Command mode):

Mcommand$ Define remainder of command line as a macro command.

Suppose you wish to invent a new Immediate mode command to move the text pointer to the end of the current line. This can be defined in terms of two Command mode commands:

```
+1L     Move to start of next line
-1C     Move back one character
```

You could of course go into Command mode and give these two commands each time you wanted to move to the end of a line (or more simply, type the two Immediate mode commands CTRL N followed by CTRL V, which have the same effect) but this quickly becomes tiresome.

 Copyright May 1980

Instead, enter Command mode by typing ESC and then type:

```
*M+1L-1C$$
```

thus defining a macro to do this. Return to Immediate mode by typing the ESC key twice and type CTRL X to execute the new command.

Try this out with a few lines of text in the text buffer. It does indeed move the pointer to the end of the current line but when the pointer is already at the end of the line it does nothing. A more subtle approach would be better: 'move the pointer to the end of the current line, or to the end of the next line if it is already at the end'. (The CTRL G command works rather like this in the opposite direction.) A suitable definition in terms of Command mode is:

```
+1C       Move forward one character
+1L       Move to start of next line
-1C       Move back one character
```

Return to Command mode and redefine the macro:

```
*mc1-c$$
```

Notice that you needn't specify the ones and the plus signs as these will be assumed, and that commands may be entered in Command mode in upper or lower case.

Try out the new command by typing CTRL X a few times. Its behaviour is much improved.

This exercise illustrates that the CTRL X command is a user definable command, and that its definition is easy (and quick) to change once you understand Command mode. Indeed this facility is perhaps the main reason for bothering to learn the Command mode editing commands.

(If you like this new command, refer to Appendix E for details of how to add it permanently.)

We could just as easily define a macro to change 'Smith' to 'Smythe':

```
*MFSmith$=XSmythe$$
```

Referring back to the problem discussed in Chapter 6, of changing all the occurrences of one character string in a document to another, the macro facility allows you to define a single character command to make the substitution. This is a particularly nice way to work because you are able to step through a document or program making the changes just by typing CTRL X, at the same time monitoring that all is well in the text window. On editors without this facility it is all too easy to perform a global string search and substitution, forgetting that the string you are changing can also occur as a substring in a longer word (e.g. Smithsonian).

MULTIPLE EXECUTION OF A MACRO

A macro can also be executed from Command mode:

nEM Execute macro command n times.

This command is equivalent to typing CTRL X 'n' times. For example, the commands:

```
*MFSmith$=XSmythe$$
*5EM$$
```

would change the next five occurrences of 'Smith' to 'Smythe'. The special symbol * can be used to specify a very large n (it is equivalent to 32767):

```
**EM$$
```

This would change all the occurrences of 'Smith' to 'Smythe', starting from the text pointer.

The text pointer is updated after each execution of a macro command and at the end of command such as the previous one, when the string search fails, the pointer is left at the position of the last successful match.

COMPARISON OF ITERATION BRACKETS AND MACROS

It can now be seen that there are three ways of replacing one string with another:

1. Repeated typing and execution of the compound command FSmith$=XSmythe$$.
2. Enclosure of the compound command within iteration brackets, allowing it to be executed a specified number of times, and
3. Saving of the command as a macro with later single or multiple execution.

The method you choose in a given situation will depend both on the circumstances and on your experience.

In general macros are safer than iteration brackets; in particular, a macro command can be stored and then tested by executing it once before letting it loose on the whole text. In contrast, a command within iteration brackets requires much more confidence. A small typing error can easily alter the text buffer beyond recognition.

Iteration brackets are useful when the Macro/Save buffer is being used for something else, for example during formatting or when an S or G command is involved.

 Copyright May 1980

ADVANCED EDITING

Finally we present two more complex examples of the use of macros. These illustrate how the text marker and macro commands can be combined to perform tasks that would normally require you to write a special program. We will drop the convention of underlining your input to the computer and add spaces and new lines within the macro definitions. These make the macros easier to read and are ignored during execution except within text string arguments.

TWO COLUMN LISTING

The following macro can be used to print a piece of text as a two column listing, saving time and paper when the list is printed. It can be applied, for example, to a list of names or a BASIC program.

In the example it is assumed that the original lines are shorter than 40 characters and that 56 lines of text are to be printed on a 72 line page, with 8 blank lines at the top and bottom.

The following macro is defined:

```
Command          Comment

8<I              Insert 8 blank lines
$>
-8P              Print 8 blank lines
56<              For 56 lines:
L -C 40<I $>     Append 40 spaces
0L 40C K I       Truncate to 40 characters
$ -L
56L I$           Go down 56 lines, mark start
L -C I$          and end of line for moving
-55L -C          Go up 56 lines, to end of line
QM               Insert second column
0P P             Print line
L                Go down one line
>                and repeat.
48K              Delete 48 blank lines, leaving 8
8P               Print 8 blank lines
8L               Go down 8 lines
$
```

A two column page is printed each time this macro is executed, and the text pointer is left at the start of the next page. A problem can arise during the printing of the last page if it is not an exact multiple of 112 lines (twice 56). The simplest solution is to append 112 blank lines to the text buffer before starting, by a command such as:

```
Z 112<I
$>$$
```

The macro can be interrupted if anything goes wrong by typing CTRL Q, since this interrupts the P command.

 Copyright May 1980

EDITING A COMPLEX MACRO

A complex macro such as this is not easy to type in correctly and will typically require several attempts to get right.

The fact that the Macro and Save buffers share the same area of storage can be used to advantage. A macro such as the one in the previous example can be typed into the text buffer just as if it was a piece of text and then saved in the Macro/Save buffer by the S command, ready for execution. The G command will 'unsave' it at the position of the text pointer for further editing. Refer to Chapter 10 for details of the S and G commands.

Note that ESC characters within the macro are held literally, not as dollar signs (from which they are indistinguishable except by TXED). To insert an ESC character into the text buffer, type CTRL ↑ ESC.

Once the macro is working correctly it can be copied into the text buffer and saved in a file for reuse.

STANDARD LETTERS

A standard letter is one which you want to send to a list of people. A secretary of a committee might wish to write to all the members telling them the date of the next meeting. It would be nicer for the members to receive a personally addressed letter than a circular.

TXED can be used to prepare such letters. We make no claims that it does so especially well, merely that you can do it. If you have a particular need to produce standard letters, you will find a number of programs advertised in the personal computer press that cater particularly for this requirement.

To produce a standard letter, you will need two files. One contains a list of names and addresses in the form:

```
&R F Hawkins
Elm Cottage
Grove Lane
WINDSOR
Berks


Dear Ralph,
&&H F Mills
156 Bath Road
MAIDENHEAD
Berks



Dear Hugh,
&
```

There are two special requirements for this file:

1. Each entry must have the same number of lines (8 in the example).

2. The entries are flagged by a special flag character (& in the example). The first entry is preceded by one flag character and the last is followed by one. Otherwise the entries are separated by two flag characters. These characters must occur nowhere else in the file.

The flag characters will be replaced by text markers before the letters are printed but since markers are removed when text is written to a file, the marker characters in the file must be something different.

The second file contains the formatted text of the letter and is produced by the QW command. A 'dummy' address is used which must have the same number of lines as the file entries. If you will be using fan-fold paper the letter should be formatted to the exact size of the paper (.PL command).

The procedure to print a series of personalised standard letters is:

1. Read in the file of addresses.

2. Change the flag characters to text markers. This can be done by repeated execution of the macro:

```
M F&$=X$$
```

3. Read in the letter.

4. Enter a macro such as:

```
M -8K QM -16P 56P$$
```

5. Move the text pointer to the beginning of the line immediately after the dummy address.

Each time the macro in step 4 is executed, the current address in the letter is deleted (-8K), a new address is filled in (QM), then one letter is printed (-16P 56P). The example assumes that the addresses occupy 8 lines and that there are 16 lines of the letter before the pointer (including the address) and 56 lines after it. You should adjust these figures to your own requirements.

The procedure works because the QM command moves the first marked block of lines it finds, searching from the beginning of the text buffer, and deletes the markers. Thus each execution of the macro moves down a new address. When all the letters have been printed the macro will fail in the QM command with the error message:

```
?Markers?
```

You can therefore start printing by the command:

```
*EM$$
```

and go and do something else until it finishes.

 Copyright May 1980

APPENDIX A

QUICK REFERENCE GUIDE

IMMEDIATE MODE

Immediate mode is entered by typing ESC ESC or an Immediate mode command immediately after the asterisk prompt. All the commands are control keys.

Moving the Pointer:

CTRL-	Chars	Words	Lines	Pages	Buffer
Forward	D	W	N	P	Z
Back	V	U	G	Q	B

Deletion:

CTRL-	Chars	Words	Lines
Forward	E		K
Back	DELT	R	

Text marker commands:

CTRL O	4.5	Insert text marker
CTRL Y ESC	4.5	Find marker

Other editing commands:

CTRL J	4.5	Insert line
CTRL S	2.7	Swap case (A-Z/a-z)
CTRL X	12.1	Execute macro
CTRL Yc	4.5	Find character c
CTRL ↑C	8.3	Insert C as control character
CTRL ↑ESC	12.5	Insert ESC

Miscellaneous commands:

CTRL A	8.1	Toggle autopaging
CTRL C	5.4	Exit to Operating System
CTRL F	8.2	Call Front Panel
CTRL I	8.2	Insert TAB character
CTRL L	8.2	Insert FORM FEED character

 Copyright May 1980

COMMAND MODE

Command mode is entered from Immediate mode by typing ESC once. Commands are executed after ESC ESC is typed.

n is a signed decimal number. If omitted, +1 is assumed. Special forms of n are:

*	9.2	Largest number (32767)
=	9.3	Minus length of last F string

Editing commands:

B	9.4	Move pointer to beginning
nC	9.5	Move pointer n characters
nL	9.5	Move pointer n lines
U	9.5	Move pointer one word
Z	9.5	Move pointer to end
nD	9.6	Delete n characters
nK	9.6	Delete n lines
Y	9.6	Delete one word
O	9.7	Swap case (A-Z/a-z)
Itext$	9.7	Insert text
↑C	9.7	Insert C as control character
nFtext$	9.8	Find text (nth match)
nXtext$	9.8	Replace n characters by text
nT	9.11	Type n lines on screen
nP	9.11	Print n lines on printer
nS	9.12	Save n lines in M/S buffer
G	9.12	Get (copy) from M/S buffer
Mcommand$	9.13	Save command in M/S buffer
nEM	9.13	Execute macro command n times
nED	9.14	Enable or disable text window
nEL	9.14	Modify text window display
nET	9.15	Set tab stops to n spaces
nES	9.15	Set screen width
nQF	9.15	Set cursor flash rate
nQS	9.16	Set automatic right margin
nQU	9.16	Define word
QX	9.16	Continue after EX
Q*	9.17	Overwrite formatter

Some useful combinations:

0PP	Print current line
B*K	Clear text buffer
Ftext$=D	Search and delete
Ftext1$=Xtext2$	Search and replace

 Copyright May 1980

Text markers:

CTRL O	4.5	Insert marker (Immed. mode)
I$	9.9	Insert marker
CTRL Y ESC	4.5	Find marker (Immed. mode)
F$	9.9	Find marker
QC	9.9	Copy marked text
QD	9.9	Delete markers after cursor
-QD	9.9	Delete markers before cursor
0QD	9.9	Delete all markers
QK	9.10	Delete marked text
QM	9.10	Move marked text

Opening and closing files:

TXED fsp	3.2	Edit or create file
EBfsp$	10.2	Edit a file
EX	10.3	End an editing session
EI	10.4	Initialise after disc change
ERfsp$	10.4	Open file for reading
EWfsp$	10.5	Open file for writing
EF	10.6	End (close) file

Reading and writing to files:

R	10.6	Read from input file
N	10.7	Write out buffer, read next
nW	10.7	Write n lines and delete them

Utility commands:

QEfsp$	10.9	Erase file
QRnewold	10.9	Rename file
QLfsp$	10.9	List file directory on screen

File specification:

A file specification (fsp) is of the general form:

[dev:]name[.type]

where name and type are 1 to 8 and 0 to 3 characters from the sets A-Z, a-z, 0-9 and $ (dollar). dev: is usually a disc name (e.g. A: to D:) but the names CON:, TTY:, LST:, LP:, RDR: and PUN: are also allowed (see Chapter 10). If dev: is omitted the logged in disc is assumed.

 Copyright May 1980

FORMATTING

QT	9.11	Output formatted text to printer
QP	9.11	Output formatted text to printer with pause
QV	9.11	Verify formatting on screen
QW	10.8	Save formatted text in file

Command	Page	Break	Initial	Default	Purpose
.LMn	11.6	Yes	12		Set left margin
.LLn	11.6	No	61		Set line length
.PLn	11.6	No	72		Set page length
.TSn	11.7	No	8		Set top space
.BSn	11.7	No	8		Set bottom space
.HSn	11.9	No	4		Set heading space
.MH////	11.8	No	None	Blank	Define heading
.FSn	11.10	No	0		Set folio space
.MF////	11.9	No	Page no	Blank	Define folio
.NCc	11.10	No	#	None	Define number character
.OP	11.8	No	Off		Omit page numbers
.PNn	11.8	No	1	1	Set page number
.FJ	11.11	Yes	On		Fill and justify
.RR	11.11	Yes	Off		Ragged right margin
.LI	11.12	Yes	Off		Literal mode
.SS	11.13	Yes	On		Single line spacing
.DS	11.13	Yes	Off		Double line spacing
.LSn	11.13	Yes	1		Set line spacing
.BR	11.13	Yes			Break line filling
.SPn	11.14	Yes		1	Space n lines
.PPn	11.15	Yes	6	Last n	New paragraph
.BPn	11.16	Yes		+1	Begin a new page
.CPn	11.16	(Yes)			Conditional new page
.PAn	11.16	Yes		1	Advance n pages
.INn	11.17	Yes	0	0	Indent subsequent text
.TIn	11.17	Yes	0		Temporary indent
.CI	11.18	Yes			Cancel indent
.CEn	11.19	Yes		1	Centre n lines
.IFn	11.20	Yes		1	Indent fully n lines
.PCc	11.20	No	None	None	Define pad character
.IG	11.21	No			Ignore comments
.ST	11.22	No			Stop previous command
.CCc	11.22	Yes			Output control character
.BE	11.23	Yes			Begin formatting long file
.EN	11.24	Yes			End formatting long file
.QY	11.25	No	On		Process daisy characters
.QN	11.25	No	Off		Ignore daisy characters
.UCc	11.25	No	None	None	Underline character
.BCc	11.26	No	None	None	Bold face character
.E2c	11.27	No	None	None	Superscript character
.E3c	11.27	No	None	None	Subscript character
.E1c	11.27	No	None	None	ESC character
.E4c	11.27	No	None	None	BACKSPACE character
.E5c	11.27	No	None	None	Overprint character

 Copyright May 1980

APPENDIX B

ERROR MESSAGES

***** IMPORTANT *****

At the start of this Appendix we describe some disc error conditions that can arise from time to time. Be prepared! If you read through this section before one of these errors occurs, you will know what to do.

The error messages that can appear during operation of TXED are described in this Appendix. Errors fall into three categories, those detected by the Operating System, Warnings, and Editing Errors.

Despite being bad practice, many people use the text editor for long periods without saving their work. Usually this is safe; the TXED program and the 380Z computer are reliable. But when a serious error does occur they stand to lose many hours of work. One of the aims of this Appendix is to help you to recover from this situation; the three more serious disc error conditions ('read only', 'bad sector' and 'disc full') are highlighted by being enclosed in boxes.

We make no excuses for reminding you:

Save your work frequently
Always make a second copy of it on a separate disc

OPERATING SYSTEM ERROR MESSAGES

Two error conditions that can arise during use of TXED are trapped by the disc Operating System. They are accompanied by unfriendly error messages of the form:

```
BDOS ERROR ON A: R/O
BDOS ERROR ON B: BAD SECTOR
```

These are referred to as 'Read Only' and 'Bad Sector' errors respectively.

READ ONLY ERROR

This error occurs when you have forgotten to give the EI command after changing a disc during an editing session. It appears when you try to open a file for writing on the changed disc (for example, by the EB or EW commands).

If you type anything other than CTRL F at this point you will lose the contents of the text buffer.

 Copyright May 1980

The recovery procedure is:

1. Type CTRL F Enter Front Panel.

2. Type J>103 and then RETURN.

This procedure restarts TXED at its 'recovery' address, 103 hex. TXED will print 'OK'.

Prevention is better than cure. Always give the EI command after changing a disc during editing (see Chapter 10).

BAD SECTOR ERROR

Although very reliable, floppy discs do wear out and the data that is stored on them in 'sectors' can no longer be read. This is detected as a 'bad' sector.

The bad sector error can occur during reading or writing to a file. If it occurs while starting an editing session, type CTRL C which will return you to the Operating System. If the error message appears while writing to a file and you have work in the text buffer that you would like to keep, proceed as follows:

1. Type CTRL F Enter Front Panel.

2. Type J>103 and then RETURN. This restarts TXED which prints 'OK'.

3. Change the bad disc

4. EI$$ Reset Operating System.

5. EWfsp$$ Open file for writing.

6. EX$$ Write text buffer to file, close file.

In either case put the suspect disc aside for further investigation.

 Copyright May 1980

WARNING MESSAGES

The following messages are output by TXED to warn you that something has happened or to give you a chance for second thoughts:

CTRL C - are you sure (Y/N)?

Appears in Immediate or Command mode when CTRL C is typed. The CTRL C command aborts an editing session, returning to the Operating System. Type Y or y if you really wish to do this.

Format error

Non-fatal error that occurs during formatting after a QT, QP or QW command has been given. An unrecognisable format command has been encountered.

Front Panel check (Y/N)?

Appears in Immediate or Command mode when CTRL F is typed. The CTRL F command allows you to enter the 380Z Front Panel. This is useful when recovering from a 'BDOS ERROR' (see above) or when wishing to select a printer option (see Appendix F). Type Y or y if you really wish to enter the Front Panel.

Lose Text Formatter (Y/N)?

Appears after the Q* command. Type Y or y if you wish to overwrite the text formatting portion of TXED. The Q*, QT, QP and QW commands will no longer be recognised if you do.

More on Input

Occurs during the R command or during the read operation implicit in the EB or N commands. Warns you that not all of the input file has yet been read. Refer to Chapter 10 for further details.

New file

Appears with the title after starting TXED with the command:

```
TXED fsp
```

Indicates that the file specified by fsp did not exist. A new file of that name is created.

Not enough memory

Fatal error that can occur immediately after TXED has been loaded. Indicates that too little memory is available for TXED's buffers. Press the RESET button and try again.

Not found

Occurs after a QLfsp$ command to list a disc directory. Indicates that no file that matches fsp has been found.

OK

Printed after TXED is restarted at address 103 hex (see above).

ERRORS THAT OCCUR DURING COMMANDS

The majority of the error messages in this group appear only in Command mode; a few may appear in Immediate mode as well. If an error occurs during a compound command in Command mode, any remaining commands are not executed.

The error message is followed by the context in which the error occurred; this can help to identify its cause. If it occurred during execution of a macro command, the words 'in macro' are appended.

?Bad argument?

Indicates an illegal or incorrectly typed numeric argument, for example after 5G$$ (n must be +1 in the G command).

?Bad command?

The command TXED is attempting to process is unrecognisable.

?Bad dev?

An illegal device has been given in a file specification. Check the characters before the colon (:). The list of permissible device names will be found at the beginning of Chapter 10.

?Bad file spec?

An illegal file name or file type has been given in a file specification. Only the characters A to Z (and a to z), 0 to 9, the dollar sign ($), and at most one full stop are allowed. Further details are given at the beginning of Chapter 10.

?Cannot close?

An error occurred during a file close operation initiated by an EX, EW or EF command. The most likely cause is that you have changed a disc before closing the file. Some data on the new disc may have been overwritten.

?Can't find?

The text string specified in an F command cannot be found. Perhaps the text pointer is nearer to the end of the text buffer than the string you are looking for.

?Dir full?

May appear (rarely) during an operation that opens a file for writing (EB, EW, TXED fsp) or that writes to a file (EX, N, W, EF). Indicates that the disc directory is full up. If the error occurs during a file opening command, delete some unwanted files with the QE command and try again. If it occurs during a file write operation, proceed as for the 'Disc full' error.

?Disc full?

May appear during any command that writes data to a file (EX, N, W or EF). All the free disc space has been used up.

IMPORTANT: TXED has automatic mechanisms to prevent loss of text under these circumstances. The following things occur:

1. The contents of the text buffer are not deleted (as they would usually be after a write operation).

2. The output file is closed.

3. If an edit backup operation is in progress (TXED fsp or EB command), the renaming of files does not take place. The input file will have its original name and the output file will be of file type .$$$. (Edit backup is explained in Chapter 5.) If you are using ER and EW commands, the files will have the names you have specified.

4. The output file will not contain all your text.

   If you have not done much editing of the file during this session, you may choose to leave TXED by the CTRL C command, delete some files to make room and start again. However recovery is possible if much editing has been done. The recovery procedure depends on whether the file that you have been editing is small enough to fit into the text buffer completely.

If it is (the most common case) all your text will still be in the text buffer. There are two possibilities. The first is to make room on the current disc by deleting files:

1.	QEfsp$$	Delete unwanted file.
2.	Repeat step 1	Until you judge you have created enough room.
3.	EWnewname$$	Create a new file for output.
4.	EX$$	Save contents of text buffer.

The second possibility is to change the full disc for one that has enough room on it:

1.	Change disc	
2.	EI$$	Reset Operating System.
3.	EWfsp$$	Create file for writing.
4.	EX$$	Save contents of text buffer.

Step 2 is vital, otherwise you will get a 'Read Only' (R/O) error (see beginning of this Appendix).

Recovery from the 'Disc full' error is a little more complicated if it occurs while editing a large file (one that fills the text buffer several times over). In this case, the text up to and including part of the present contents of the text buffer will have been saved in the current output file. As noted above, this will have file type .$$$ if an edit backup operation is in progress. The remainder of the file consists of part of the text buffer plus whatever has not yet been read from the input file. The safest procedure is as follows:

1.	Change disc	
2.	EI$$	Reset Operating System.
3.	EWfsp$$	Create file for writing.
4.	B$$	Move pointer to start of text buffer.
5.	*W$$	Save contents of text buffer.
6.	EF$$	Close output file.

 Copyright May 1980

Your original file is now split across three files. The first (edited) portion is in the file that was being written when the error occurred. The second portion (which will partially overlap the first) is in the file you have just created. The third portion is that part of the input file that has not yet been read. These three portions can now be combined, deleting the overlapping bits.

?"EB" already active?

Warning which appears when an EW command is attempted while an edit backup operation is in progress. The EW command is ignored. Complete the backup operation with the EX command.

?"EB" - more input?

Warning which appears when an ER command is attempted while an edit backup operation is in progress and there is still some text to be read from the input file. The ER command is ignored. Complete the reading of the input file with an R, N or EX command, then try again.

?End of file?

Indicates an attempt to read past the end of the input file. Appears when an R or N command is given after the whole of the input file has been read. This message is harmless and it is sometimes reassuring to evoke it with an R command when you can't remember whether all the input file has been read in.

?Error in Editor?

Indicates an error in TXED's logic. Please report the circumstances to Research Machines. Could also indicate a hardware fault.

?Display error?

Appears during the long line display mode set by the EL$$ command. Indicates that attempts to display very long lines have failed. TXED reverts to short line mode, as set by the 0EL$$ command (see Chapter 9).

?File exists?

Appears when you specify a file that already exists in an EW command. The EW command is ignored. If you really wish to create a file of that name, erase the existing one with

the QE command and try again.

?File not found?

Appears when a file specified by an ER or EB command cannot be found.

?File not open?

Occurs if the QW command is given and no file has been opened for writing. Open a file and try again.

?Illegal extension?

Appears if you specify a file type of .$$$ or .BAK in an EB command or when starting the editor by the TXED fsp. These file types may not be specified for an edit backup operation. Extension is another name for file type.

?Iteration fault?

Occurs during execution of a compound command that uses iteration brackets (<>) if there are an unequal number of left and right brackets. The command is aborted.

?Markers?

Appears during execution of a QM or QC command if the text pointer is within the marked block. See Chapter 4.

?No file?

A file read or write operation has been attempted but the relevant file has not been opened. Open a file and try again.

?No room?

Occurs during an insert operation (such as typing in text in Immediate mode) or during an QC command. Indicates that the text buffer is nearly full. The text buffer contents should be written to the output file.

?Output file open?

Warning which appears when an EB command is attempted while a file is open for output. Close the current output file with an EF command and try again.

 Copyright May 1980

?Read error?

Should not occur during normal operation (but see beginning of this Appendix).

?Write error?

Should not occur during normal operation (but see beginning of this Appendix).

 Copyright May 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

APPENDIX C

GLOSSARY

ARGUMENT

A term given to a part of a command which can vary and which affects the action of the command. In Command mode, commands may optionally be preceded by a numeric argument and some can be followed by a text string argument. Many formatting commands are followed by an argument, usually numeric.

ASCII

American Standard Code for Information Interchange. The form in which characters are encoded within the computer.

BOOTSTRAP

A term used to refer to the process of loading the Disc Operating System into memory. The first part of the System loads in the remainder. The process has been compared to pulling oneself up by one's bootstraps.

BUFFER

A part of computer memory set aside for storage of text or editing commands. It can be filled or emptied without disturbing the contents of the rest of memory. See text buffer and Macro/Save buffer.

CLOSE

A process applied to a file that is open for writing, in order to terminate it correctly and update its directory entry.

COMMAND MODE

Mode used for the more complex editing commands and indicated by an asterisk (*) prompt. Commands are entered on the bottom four lines of the screen and usually consist of more than one character. Functions include file handling, printing, searching for text strings and movement of blocks of text.

COMMAND LINE

That sequence of commands, text strings, file specifications, etc. entered from the keyboard and executed after RETURN in CP/M or after two ESC characters in TXED.

 Copyright May 1980

COMMAND TERMINATOR

In Command mode, the ESC character which terminates the text string argument that can follow certain commands. Two ESC characters typed in succession cause any preceding commands to be executed.

CONTROL CHARACTER

A character typed while the CTRL key is depressed. The name originates from devices such as the teletypewriter which uses these characters to perform control functions such as turning on and off the paper tape reader. In TXED, control characters typed at the keyboard perform editing functions in Immediate mode. Certain formatting commands can send control characters to the printer to perform specific functions.

COS MONITOR

Abbreviation of Cassette Operating System for 380Z, used also as the lowest level of the Disc Operating System.

CP/M

Abbreviation of Control Program for Microcomputers. An implementation of the Digital Research program used as the 380Z Disc Operating System.

DEFAULT

The value of a numeric or text string argument that may be assumed by the computer when no value is supplied.

DEVICE NAME

A 1 to 3 character name, preceding a file name and separated from it by a colon, used to indicate the device upon which input or output is to take place. A list of the permissible names will be found at the beginning of Chapter 10. If omitted, the logged in disc (disc version) or cassette (cassette version) is assumed.

DIRECTORY

A special area of a disc that contains a list of the names of the files stored on that disc, together with information about where to find them. The list of file names can be printed by the QL command.

 Copyright May 1980

EDIT BACKUP

The process of editing a file while retaining its state immediately prior to editing. The file type of the original file is changed to .BAK to indicate the back up copy and the original file name is transferred to the edited version.

ERROR MESSAGE

Response on screen to a faulty command, or an error during transfer to or from a file. See Appendix B.

EXTENSION

Another name for file type.

FILE

A file is the form in which a program or section of text is stored on disc or cassette tape. See Chapters 3 and 10.

FILE NAME

The middle and essential part of a file specification (sometimes also referred to as the primary file name). Usually gives an indication of the file's purpose. See Chapters 3 and 10.

FILE SPECIFICATION

Comprises a disc or device name, a file name and an optional file type, and uniquely identifies a file. Abbreviated to 'fsp'. See Chapters 3 and 10.

FILE TYPE

The last part of a file specification, separated from the preceding file name by a full stop (sometimes also referred to as the secondary file name or extension.) Usually gives an indication of the file's contents. See Chapters 3 and 10.

FOLIO

The line of characters at the bottom of a page which contains the page number. TXED allows up to three elements to be specified in a folio. See .MF command, Chapter 11.

 Copyright May 1980

FORMATTING

The process of neatly laying out a document on a suitable printer. See Chapter 11.

FORMATTING COMMAND

Formatting commands are embedded within the text and identified by the occurrence of a full stop in the first character position on a line. They consist to a two letter command, possibly followed by an argument, usually numeric.

IMMEDIATE MODE

Mode used for text entry from the keyboard and simple editing using commands consisting of a single control character. Indicated by an exclamation mark prompt (!).

INPUT FILE

The file from which TXED will read text into the text buffer.

ITERATION

Automatic repetition of a command, often by using iteration brackets (<>). See Chapter 9.

LINE

Within the text buffer, a series of characters up to and including a RETURN character. When text is written to a file, a LINE FEED character is added after each RETURN.

LISTING

Refers mainly to the text of a program or document as listed on the printer, but applied colloquially when the same appears on the screen.

LOGGED IN DISC

A disc unit is logged in to CP/M by typing a command line consisting only of its name. For example, the command 'B:' logs in to disc B. The name of the logged in disc is used as a prompt in CP/M. A file specification which does not contain a device name is assumed to refer to a file on the logged in disc unit.

 Copyright May 1980

MACRO COMMAND

Stored series of one or more commands which can be executed subsequently by a single command. The command is saved in the Macro/Save buffer which itself can be edited. See Chapter 12.

MACRO/SAVE BUFFER

A small buffer within TXED used for storage of a macro command, saving small portions of text, and during formatting. It can only be used for one of these purposes at a time.

OPEN

The process of making a file ready for reading or writing.

OUTPUT FILE

The file to which TXED will write text from the text buffer.

PAGE OF INPUT FILE

A series of lines of text, up to and including a FORM FEED (CTRL L) character. Reading of the input file stops when a FORM FEED is encountered.

PIP

The Peripheral Interchange Program, used mainly for copying files from disc to disc and for listing them on the printer.

PROMPT

Indication on the screen that the computer is ready for you to type a command.

READING

The process of transferring data from a file to the computer.

RESTART ADDRESS

The memory address to which execution can be transferred to restart TXED without losing the contents of the text buffer.

 Copyright May 1980

STAT

The File Status program that displays the size of files and the amount of free space on a disc.

TAB

A single character (CTRL I) which is expanded when displayed on the screen or sent to the printer to enough spaces to reach the next tab stop on a line. Unlike a typewriter, tab stops occur at equal intervals of 1, 2, 4 or 8 spaces. See ET command, Chapter 9.

TEXT BUFFER

The area within the computer that is filled by reading from the input file, and in which all editing takes place. Text can also be entered into the text buffer from the keyboard. When editing is complete, the contents of the text buffer are written to the output file.

TEXT MARKERS

Non-printing signals inserted into the text to permit the marked point to be found again later, or to allow a block of text delimited by text markers to be moved, copied or deleted. Used also in formatting for selective output.

TEXT POINTER

Editing of text takes place in relation to the position of the text pointer within the text buffer. The pointer lies between two characters but is normally shown on the screen overlying the character following its position.

TEXT STRING

A sequence of characters.

TEXT WINDOW

The upper part of the screen containing the cursor that indicates the position of the text pointer and displays the contents of the text buffer around the pointer.

WRITING

The process of transferring data from the computer to a file.

 Copyright May 1980

APPENDIX D

MAKING A WORKING DISC

The disc on which you receive TXED from Research Machines should not be used routinely. The programs on it should be copied to a working disc and the Distribution disc put away in a safe place. You may need it if your working disc becomes damaged or lost, or, if an upgrade becomes available, to return to Research Machines as proof of purchase to qualify for a reduced price for the upgrade.

The procedure for making a copy of the disc is outlined in the 380Z System Manual. It will be repeated here for convenience. The description assumes a system with two disc drives. If using an MDS system with a single drive, refer to the System Manual.

1. Format a blank disc using the program FORMAT provided on the CP/M System disc. With a copy of the CP/M System disc in unit A and the blank disc in unit B, type:

   A>FORMAT

   followed by the RETURN key, then follow the instructions given by the program.

2. Reboot the Operating System, then copy the CP/M Operating System to the newly formatted disc using the program SYSGEN, also present on the System disc. Type:

   A>SYSGEN

   then RETURN. Answer A to the first question and B to the second. When the operation is complete, reboot the Operating System by typing RETURN.

3. Copy all the programs from the TXED Distribution disc to the new disc using the PIP program on the System disc. Type:

   A>PIP

   then RETURN. When the asterisk prompt appears remove the System disc from unit A and replace it with the TXED disc. Type:

   B:=A:.*[V]

   then RETURN. This will copy all the programs from the TXED disc to the working copy, listing their names on the screen as it does so.

4. Test the new working disc. Take the TXED Distribution disc out of drive A and put it away in a safe place. Move the working disc from drive B to drive A. Press the RESET button and boot CP/M by typing B.

 Copyright May 1980

There will probably only be one or two programs on the Distribution disc, one of which will be TXED.COM. This is the editor. You may like to rename it to EDIT.COM, using the CP/M REN command:

```
A>REN EDIT.COM=TXED.COM
```

You will now be able to type 'EDIT' or 'EDIT filename' when you want to start an editing session.

 Copyright May 1980

APPENDIX E

CONFIGURING TXED

This Appendix describes how to configure TXED for personal use.

A number of the options that can be altered by means of commands given in Command mode (for example, cursor flash rate) can be made permanent by saving a copy of the editor with the preferred options selected. On reloading the new copy, the selected options will be in force.

We suggest that you call your personal version of the editor EDIT.COM. You will then be able to type the command 'EDIT fsp' which is a natural way of working.

More far reaching configuration is possible by 'patching' the editor, that is, by changing its code. Patching is carried out by means of the 380Z Front Panel and can be used to change the initial settings of the formatter, the Immediate mode commands, and some other items as described below. You should not attempt to patch the editor until you have had some experience of using the 380Z Front Panel facilities.

CONFIGURATION BY COMMAND

The commands described in the Utility Commands section of Chapter 9 select various options within TXED that control its behaviour. If a copy of TXED is saved with the preferred options in effect, these options will be in force when the copy is reloaded.

Suppose you prefer the displacement cursor of previous versions of TXED to the flashing one of the current version. If you select the displacement cursor by giving the 0QF command and then save a copy of TXED with the cursor in this state, the displacement cursor will be selected when the new copy is loaded. This new copy becomes your personal version of TXED.

The commands that fall into this category are:

nED	Enable or disable text window.
nEL	Select text window display mode.
nES	Set screen width (80 or 40 characters).
nET	Set TAB character expansion.
nQF	Set cursor flash rate.
nQS	Set word wrap margin.
nQU	Select word definition.
QX	Continue after EX.
Q*	Disable formatter.

 Copyright May 1980

Refer to Chapter 9 for more information about these commands. All commands except the last two merely select one of a number of options and the settings to choose are the ones you use most frequently. The last two commands effect a permanent change, that is, you cannot reverse the QX command and you cannot put back the formatter (except by loading a copy of TXED that includes it).

The two versions of TXED that are distributed, the one with formatting facilities and the other without, are in fact copies of the same version, except that in the case of the second, the Q* command was given before making the master copy. If you have purchased the version that includes the formatter, you may wish to generate a non-formatting version as well for editing programs. This will have a larger text buffer and will be quicker to load into memory.

SAVING A CONFIGURED VERSION

Once you have configured TXED, by giving commands or by patching, you will need to save a copy of it in the configured state using the CP/M SAVE command. It is advisable to give the copy a new name as the SAVE command overwrites any existing file of the same name.

As an example, the full procedure for changing the initial state of the cursor to a displacement cursor is:

1. TXED — Load TXED.
2. 0QF$$ — Set displacement cursor.
3. CTRL C — Leave TXED (type Y in reponse to query).
4. SAVE nn EDIT.COM (see below)

The 'nn' in the SAVE command is the number of 256 byte 'pages' of memory to save. This must be given as a two digit decimal number. The number of pages you specify will depend on whether or not you are saving a version that includes the formatter. The values to use will be found in the Release Note at the beginning of this Manual.

Note that there is a problem in CP/M Version 1.4 that corrupts the memory image at the end of the SAVE command. If you give the SAVE command twice in succession (e.g. to make a backup copy) the second copy will be incorrect.

CONFIGURATION BY PATCHING

It is also possible to make changes in TXED by patching (or modifying) the code. A number of default options can be changed. In addition you can change the Immediate mode command table and alter the initial values assumed by the formatter. The following is an excerpt from the beginning of the Assembly Language listing. Users of C4100 cassette systems (COS 2.2 and 2.3) should add 4300 hex to all the addresses.

 Copyright May 1980

```
0100 C38C06             JP      EDITOR  ;Cold start
0103 C3E306             JP      RESTRT  ;Restart
0106 161D       PTEX:   DEFW    DEFBASE ;Pointer to formatter table
0108 F027       PFIN:   DEFW    TXFIN   ;Pointer to end of editor
010A 7601               DEFW    IML     ;Pointer to Immediate mode table
010C 4002               DEFW    IOP     ;Pointer to end of table space + 1
```

Address 106 hex contains a pointer to the table of initial values used by the formatter. Thus in this example the formatter table starts at address 1D16 hex. The start of this table may be different in your version of TXED but its address will still be contained in address 106 hex.

Similarly, address 108 hex contains a pointer to the end of the editor, address 10A hex a pointer to the start of the Immediate mode table, and address 10C hex a pointer to an address one past the end of the space available for the table of Immediate mode commands.

PATCHING THE INITIAL FORMATTER SETTINGS

As mentioned in the previous section, address 106 hex contains a pointer to the table of values used by the formatter to set up its initial parameters. The following is a listing of the table:

```
                ;Default values
1D16 3D         DEFBASE:DEFB    61      ;Line length
1D17 00                 DEFB    0       ;Indent
1D18 00                 DEFB    0       ;Mode
1D19 06                 DEFB    6       ;Paragraph indent
1D1A 01                 DEFB    1       ;Line spacing
1D1B 01                 DEFB    1       ;Gap direction
1D1C 48                 DEFB    72      ;Page length
1D1D 08                 DEFB    8       ;Top margin
1D1E 08                 DEFB    8       ;Bottom margin
1D1F 04                 DEFB    4       ;Header margin
1D20 0C                 DEFB    12      ;Page offset
1D21 0100               DEFW    1      ;Page number (word)
1D23 00                 DEFB    0       ;Line number
1D24 00                 DEFB    0       ;Literal space mapping
1D25 00                 DEFB    0       ;Underline character
1D26 00                 DEFB    0       ;Output flip flop on
1D27 00                 DEFB    0       ;Output flag on
1D28 00                 DEFB    0       ;Folio margin
1D29 23                 DEFB    '#'     ;Page number character
1D2A 00                 DEFB    0       ;Bold face character
1D2B 00                 DEFB    0       ;Underline flip flop
1D2C 00                 DEFB    0       ;Bold face flip flop
1D2D 00000000           DEFB    0,0,0,0,0 ;Escape sequence chars 1-5
                DEFBF:                  ;End of list
                ;--------
                ;Pointers to special sequences
1D32 471D       PULS:   DEFW    ULS
1D34 4F1D       PBF1:   DEFW    BF1
1D36 561D       PBF2:   DEFW    BF2
```

```
1D38 601D       PBF3:   DEFW    BF3
1D3A 691D       PPFS:   DEFW    PFS
1D3C 711D       PESC1:  DEFW    ESC1S
1D3E 771D       PESC2:  DEFW    ESC2S
1D40 7D1D       PESC3:  DEFW    ESC3S
1D42 831D       PESC4:  DEFW    ESC4S
1D44 891D       PESC5:  DEFW    ESC5S
                ;----------
1D46 20         FIXPAGE:DEFB    SPACE           ;Force Anadex to print
                                                ; first line of page
                ;----------
                ;Underline sequence
1D47 1B4E5F     ULS:    DEFB    ESC, 'N', 5FH
1D4A 00                 DEFB    0               ;Terminator
1D4B 00000000           DEFB    0,0,0,0         ;Spare
                ;Bold face sequences
1D4F 1B4E       BF1:    DEFB    ESC, 'N'
1D51 00                 DEFB    0               ;Terminator
1D52 00000000           DEFB    0,0,0,0         ;Spare
1D56 1B         BF2:    DEFB    ESC
1D57 48404041           DEFM    'H@@A'
1D5B 00                 DEFB    0               ;Terminator
1D5C 00000000           DEFB    0,0,0,0         ;Spare
1D60 1B         BF3:    DEFB    ESC
1D61 48504041           DEFM    'HP@A'
1D65 00000000           DEFB    0,0,0,0
                ;----------
                ;Page footer sequence
1D69 00000000   PFS:    DEFB    0,0,0,0,0,0,0,0
                ;----------
1D71 1B000000   ESC1S:  DEFB    ESC,0,0,0,0,0   ;Print ESC SPACE (pound) etc.
1D77 1B440000   ESC2S:  DEFB    ESC,'D',0,0,0,0 ;Up
1D7D 1B550000   ESC3S:  DEFB    ESC,'U',0,0,0,0 ;Down
1D83 08000000   ESC4S:  DEFB    8,0,0,0,0,0     ;BACKSPACE
1D89 1B4E0000   ESC5S:  DEFB    ESC,'N',0,0,0,0 ;No motion on next
                ;----------
1D8F 971D       VTBLD:  DEFW    VTBLDS
1D91 9F1D       VTBL1:  DEFW    VTBL1S
1D93 A71D       VTUND:  DEFW    VTUNDS
1D95 AF1D       VTUN1:  DEFW    VTUN1S

1D97 1B3D3143   VTBLDS: DEFB    ESC,'=','1','C',0,0,0,0  ;Dim on
1D9F 1B3D3043   VTBL1S: DEFB    ESC,'=','0','C',0,0,0,0  ;Dim off
1DA7 1B3D3142   VTUNDS: DEFB    ESC,'=','1','B',0,0,0,0  ;Underline on
1DAF 1B3D3042   VTUN1S: DEFB    ESC,'=','0','B',0,0,0,0  ;Underline off
                ;----------
```

Values such as the initial setting of the line length (the first item in the table) can be changed. Note that the entries in the table are in hexadecimal.

In the third entry ('Mode'), set bit 7 to inhibit page numbers (.OP) and bit 2 to turn off justification (.RR). Set bit 6 to inhibit the special daisy wheel characters (.QN). The least significant bit is bit 0.

As an example, to change the initial line length to 72, the procedure would be:

1.	TXED	Load TXED.
2.	CTRL F	Enter front panel (type Y in reponse to query).
3.	M>106 RETURN	
4.	I	Point to table.
5.	48 RETURN	72 in hex.
6.	K	Return to TXED.

You would then carry out steps 3 and 4 of the instructions for saving a configured version (see above).

The second half of the table, after label DEFBF, contains special character sequences, each terminated by a zero byte.

FIXPAGE is a sequence that is sent at the beginning of each page and resolves a problem on Anadex DP-8000 printers (see Appendix F). It is set up initially to send one space character.

PFS is a sequence, initially empty, that is sent at the end of each page. You can patch in a FORM FEED character here (0C hex) if your printer responds to one. If you do, you will have to set the logical page length with the .PL command to less than the physical page length.

ULS is the underline sequence that is sent before each underlined character. The initial setting for the Qume Sprint 5 is for there to be no motion after the next character (ESC N) which is an underline character (5F hex). The print head will not advance after the underline character is printed, so that the next character to be printed will be underlined. You will probably need to change this and the following sequences if you have a daisy wheel printer from a different manufacturer.

BF1 to BF3 are the bold face sequences that are sent before the a bold face character is printed, after it has been printed once, and after it has been printed a second time.

ESC1S to ESC5S are the strings that are output in response to the characters designated by the .E1 to .E5 commands.

VTBLDS and VTBL1S are sequences sent to an 80 character VDU board, if one is installed, before and after a bold face character. These sequences make the character dim. Inverse video is a reasonable alternative here, although the choice of dim was based on some experimentation.

VTUNDS and VTUN1S are sequences sent to an 80 character VDU board, if one is installed, before and after an underlined character. These sequences underline the character.

 Copyright May 1980

PATCHING THE IMMEDIATE MODE TABLE

Address 10A hex contains a pointer to the table at label IML which contains a series of 3 byte entries. The first byte of each entry represents an Immediate mode control character and the following two bytes point to the corresponding Command mode macro to be obeyed. The table is terminated by a zero value together with a pointer to the fall through situation which is a single character insert.

The macros consist of the commands to be interpreted, terminated by a byte containing -1 (FF hex). The first few entries of the table are treated in a special fashion and should not be modified. The remainder may be changed by the user provided the above rules are maintained.

The 2 byte address contained at address 10C hex represents the start of code beyond that reserved for the macro tables. The tables must be kept below this limit.

You may thus choose to change the control keys that are associated with each command, the commands themselves, or both.

```
                ;Immediate mode list
0176 1EDB01     IML:    +CTRLUA,#X.UA
0179 19D701             +CTRLY, #X.CY
                ;Users may modify from here on
017C 7FDE01             +RUBOUT,#X.RB
017F 020302             +CTRLB, #X.CB
0182 04E501             +CTRLD, #X.CD
0185 05DF01             +CTRLE, #X.CE
0188 07E701             +CTRLG, #X.CG
018B 0AE101             +CTRLJ, #X.CJ
018E 0BEC01             +CTRLK, #X.CK
0191 0EEA01             +CTRLN, #X.CN
0194 0F0002             +CTRLO, #X.CO
0197 10EF01             +CTRLP, #X.CP
019A 11EE01             +CTRLQ, #X.CQ
019D 12FD01             +CTRLR, #X.CR
01A0 13F601             +CTRLS, #X.CS
01A3 15FA01             +CTRLU, #X.CU
01A6 16E401             +CTRLV, #X.CV
01A9 17F801             +CTRLW, #X.CW
01AC 18F301             +CTRLX, #X.CX
01AF 1A0502             +CTRLZ, #X.CZ
                ;Expansion space and list terminator
01B2 00D301             0,      #X.IN
01B5 00D301             0,      #X.IN
01B8 00D301             0,      #X.IN
01BB 00D301             0,      #X.IN
01BE 00D301             0,      #X.IN
01C1 00D301             0,      #X.IN
01C4 00D301             0,      #X.IN
01C7 00D301             0,      #X.IN
01CA 00D301             0,      #X.IN
01CD 00D301             0,      #X.IN
01D0 00D301             0,      #X.IN
```

 Copyright May 1980

```
                ;Immediate mode macros
01D3 49001BFF   X.IN:   'I', 0, ESC, -1 ;Insert
01D7 46001BFF   X.CY:   'F', 0, ESC, -1 ;Single char search
01DB 5E00FF     X.UA:   '↑↑', 0, -1     ;Insert control char
                ;Users may modify from here on
01DE 2D         X.RB:   '-'             ;Rubout backwards
01DF 44FF       X.CE:   'D', -1         ;Rubout forwards
01E1 490D1B     X.CJ:   'I', CR, ESC
01E4 2D         X.CV:   '-'
01E5 43FF       X.CD:   'C', -1
01E7 2D4330     X.CG:   '-C0'
01EA 4CFF       X.CN:   'L', -1
01EC 4BFF       X.CK:   'K', -1
01EE 2D         X.CQ:   '-'
01EF 31354CFF   X.CP:   '15L', -1
01F3 454DFF     X.CX:   'EM', -1
01F6 4FFF       X.CS:   'O', -1
01F8 5555       X.CW:   'UU'
01FA 2D55FF     X.CU:   '-U', -1
01FD 2D59FF     X.CR:   '-Y', -1
0200 491BFF     X.CO:   'I', ESC, -1
0203 42FF       X.CB:   'B', -1
0205 5AFF       X.CZ:   'Z', -1
```

↓240H ↓(10C) = 240H

Suppose you wish to add a new command, CTRL T, that moves the cursor to the end of the line. As discussed in Chapter 12, a suitable Command mode macro is 'CL-C'. The procedure is as follows:

1. TXED — Load TXED.
2. CTRL F — Enter front panel (type Y in reponse to query).
3. M>10A RETURN
4. I — Point to table.
5. Find end of command list (1B2 hex in example).
6. Find end of macro list (207 hex in example).
7. Insert new command:

```
!M>1B2 RETURN
!14 RETURN      (CTRL T)
!07 RETURN      (address of command)
!02 RETURN
!M>207 RETURN
!43 RETURN      (C)
!4C RETURN      (L)
!2D RETURN      (-)
!43 RETURN      (C)
!FF RETURN      (-1)
```

8. K — Return to TXED.

 Copyright May 1980

You would then carry out steps 3 and 4 of the instructions for saving a configured version (see above).

Remember that the address and contents of the table may be different in your version of TXED. Use the contents of address 10A to find the table.

DEFAULT FILE TYPES

The default file types that will be used when none is specified can be found at the following addresses:

163 hex	For EB command
16B hex	For ER, EW and old file name of QR command
16F hex	For new file name of QR command

These are set initially to three spaces and are used if no file type is supplied.

To allow ambiguous file names in the QE command (at your own risk), change location 172 hex from 0 to 1.

KEYBOARDS WITH ALT MODE

Some older types of 380Z keyboard do not have an ESC key. Instead the top left key is marked 'ALT MODE'. Many of these keyboards in fact generate the ESC character when this key is pressed! If you have such a keyboard, the first thing to do is to load TXED and try typing this key twice. If the prompt character changes from an asterisk (*) to an exclamation mark (!), all is well. For ESC read ALT MODE throughout the Manual.

If the prompt character does not change, rub out whatever was printed (it should be a 3/4 sign), then change the contents of location 111 hex from 1B hex (ESC) to 7D hex (ALT MODE):

1. CTRL F — Enter front panel (type Y in reponse to query).
2. M>111 RETURN
3. 7D RETURN
4. K — Return to TXED.

Try typing the ALT MODE key again. It should print as $ and typing it twice in succession should change the prompt character. After modifying location 111 you should carry out steps 3 and 4 of the instructions for saving a configured version (see above).

CURSOR CHARACTER

The two cursor characters are stored as follows:

Address	Value	Meaning
10F	95	Normal cursor character
110	BF	Cursor before RETURN character

If you wish you may modify these according to taste. For example, values of D5 and FF give a brighter cursor.

 Copyright May 1980

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980

APPENDIX F

SETTING UP A PRINTER

This Appendix contains general information about setting up a printer option and some specific remarks about the OKI Microline 80, Anadex DP-8000 and Qume Sprint 5.

GENERAL INFORMATION

Connecting a printer to the 380Z requires that the printer be plugged into an appropriate connector on the back panel of the computer, and informing the 380Z that you have done this. Both of these operations, while being simple in principle, are very easy to get wrong, so if it doesn't work, try again after rereading the instructions.

If your printer has a parallel interface, it should be plugged into the User I/O socket on the back panel. This is a 25-way D submin female socket connected to the boards of the computer by a flat 20 way cable. It may be marked P1.

There are several types of serial interface supplied with the 380Z. Most of them appear on the back panel as 25-way D submin sockets, connected to the board by relatively few wires - 10 at most and probably less. One may be marked S1. 20 mA current loop interfaces use a 6-pin DIN connector.

If you are unsure of the type of interface on your printer, check the socket on the printer. All parallel type printers sold by Research Machines accept a plug which has a single oblong plastic tongue surrounded by 36 contacts. No serial printers use this sort of connector.

After connecting your printer to the 380Z, you must set up a printer option. This is done by means of the O command from either COS command mode or from the Front Panel. Assuming that you already have TXED loaded, the easiest way to select a printer is from the Front Panel. You can get there by typing CTRL F and answering Y or y to the question:

Front Panel check (Y/N):

The top part of the screen will be filled with a bewildering array of characters. Ignore these and concentrate on the bottom four lines of the screen. Note that the prompt character (the character in the bottom left corner of the screen) is an exclamation mark (!).

Type O (without using RETURN). On versions of COS earlier than 3.3, the message

CASSETTE OR LINEPRINTER (C/L):

will appear. Type either L or l (again, no RETURN) to indicate a printer option. Later versions of COS may not allow cassette options, so this

 Copyright May 1980

message is not printed. A message of the form

```
PRINTER TYPE (0-4):
```
or
```
Printer type (0-6):
```

is displayed. You should now enter the type of your printer. Parallel interfaces (e.g. Anadex DP-8000, Centronics) are always type 3. Printer type 0 is the screen, which means that all output sent to the printer (e.g. with the QT command) appears on the screen. The 380Z normally comes on in this state.

SIO-2 and SIO-3 serial interfaces are printer type 2. Type 4 is used for SIO-4 and SIO-1 Line Printer Option 4 interfaces. Type 1 is for SIO-1B and SIO1A Line Printer Option 1 interfaces. Printer types 5 and 6 are for additional type 4 interfaces, if you have more than one of these.

If you are unsure of the particular type of your serial interface, check the board inside the computer which connects directly to the socket on the back panel. If the board is a small one, about three inches by two, it is either an SIO-2 or SIO-3, and therefore type 2. If is is a standard sized card (about 10"x5") with about a third of the board completely blank, it is a type 1 interface. Otherwise it is type 4.

After selecting printer type 3, the message

```
Attend to printer!
```

may appear. This means that the printer failed to respond correctly to the 380Z. Usual causes are:

Printer not switched on.
Printer plugged into the wrong socket on the 380Z.
Wrong type of connecting cable used.
Printer out of paper.
Printer not selected by select/on line switch on printer.

After selecting printer types 2, 4, 5, or 6, the message

```
Baud Code (0-6):
```

will appear. The Baud rate is a measure of how fast the printer is, and the rate at which the computer sends must match the rate at which the printer expects to receive. Check the printer to find out the expected Baud rate. Then select the appropriate code from the table below.

Code	Baud rate
0	110
1	300
2	600
3	1200
4	2400
5	4800
6	9600

Printer type 2 will not go faster than 2400 Baud. Printer type 1 baud

rates are set by a small switch on the back panel.

After a printer has been selected, return to TXED by typing K (again, do not type RETURN). Test the printer by typing in one or two lines of text (making sure there is a RETURN after each line), then moving the text pointer to the beginning of the text buffer (CTRL B), typing ESC once and then the Command mode command:

*99P$$

(remembering that $ stands for ESC). If this prints nothing, then check the printer type and that you have connected the printer correctly. If it prints garbage, then the Baud rate is probably set incorrectly; go back to the Front Panel and try another.

OKI MICROLINE 80

This printer may be supplied with a parallel or serial interface. If it has a parallel interface, the connectors at the two ends of the printer cable will be obviously different and you should select printer type 3. If a serial interface, the two plugs will be similar (although one will be marked 'OKI end' or something similar) and there will be a small black box on the back of the printer. In this case you will normally select printer type 4 at 9600 Baud (code 6).

When the Microline 80 is turned on it sets itself to a horizontal spacing of 10 characters per inch and a vertical spacing of 6 lines per inch. When printing formatted text, you may wish to alter these settings. The .CC command can be used to alter the character spacing:

.CC=	16.5 characters per inch
.CC>	10 characters per inch
.CC?	5 characters per inch

The new spacing will apply until you select another. Similarly, the .E1 command can be used to alter the line spacing and logical page width:

.E1&		
&6	6	lines per inch
&8	8	lines per inch
&A	80	characters per line
&B	64	characters per line

Refer to Chapter 11 for further details of the .CC and .E1 commands.

ANADEX DP-8000

Although they possess both serial and parallel interfaces, Anadex printers are normally used with a parallel interface on the 380Z. Select printer type 3.

On early models of the Anadex there was a problem with skipping over the page perforations. The Anadex kept count of its vertical position on the page and when it considered it was at 'top of form' it ignored all blank

lines until it received a printing character. This played havoc with attempts to send it formatted text. If you have such problems, first examine the Anadex switch settings, making sure that they are set for the length of paper you are using (normally 12 inches) and that the 'skip over perforation' feature is turned off. As a last resort, refer to Appendix E and try changing the top of page character from a space character (which cures the problem on most Anadex printers) to a printing character.

The .CCN command will cause the printing of double width characters for one line.

QUME SPRINT 5

The Qume Sprint 5 printer requires a serial interface; this can be run at rates from 110 to 1200 Baud (codes 0 to 3). We recommend that you use 1200 Baud if possible as this gives considerably higher throughput, the printer being able to optimise its print head movements to a considerable degree. This rate requires that the 'Ready' line be connected on the interface. If you have problems running at this rate, contact Technical Support.

 Copyright May 1980

APPENDIX G

DIFFERENCES FROM TEX

This Appendix describes the differences between the TXED formatter and the Digital Research program TEX that was formerly sold by Research Machines for text processing.

The TEX name is a trademark of Digital Research. Research Machines' TXED has been in existence longer but has only recently had formatting facilities added to it.

The TXED formatter implements all the commands described in the TEX User's Guide as well as a number of new commands. Some of the commands have different names but all the TEX names are recognised as well. This is to allow document files that have been prepared for TEX to be formatted by TXED with few changes.

The correspondence between TEX and TXED command names is:

TEX	TXED	Function
.AD *	.FJ	Fill and justify
.BP	.BP	Begin page
.BR	.BR	Break
.CE	.CE	Centre lines
.CP	.CP	Conditional page
.DS	.DS	Double spacing
.HE	.HE	Heading (TXED also has .MH)
.HM *	.HS	Heading space
.IG	.IG	Ignore comments
.IN	.IN	Indent
.LI	.LI	Literal mode
.LL	.LL	Line length
.LS	.LS	Line spacing
.MB *	.BS	Bottom space
.MT *	.TS	Top space
.NA *	.RR	Ragged right
.OP	.OP	Omit page numbers
.PA	.PA	Page advance
.PL	.PL	Page length
.PN	.PN	Page numbering
.PO *	.LM	Left margin
.PP	.PP	Paragraph
.QI *	.CI	Cancel indent
.SP	.SP	Space n lines
.SS	.SS	Single spacing
.TI	.TI	Temporary indent

In the case of the commands marked with an asterisk, both forms will be recognised.

The effects of the commands are not always identical. Most of the differences are intentional and are based upon experience of using TEX.

 Copyright May 1980

The main differences are as follows:

Literal mode: the text will be offset by the current indent as set by the .IN command. TAB characters are expanded according to the current setting of the ET command, with the first column after the indent regarded as column zero. The result of this is to produce a similar appearance on the printed page and on the screen. Production of tables is simplified.

Fill and Justify: multiple spaces and TAB characters are contracted to a single space and act as a single word separator, with two exceptions. A blank line is equivalent to an .SP 1 command. A line starting with a TAB character is equivalent to a .PP command. These changes result in excess spaces being ignored and simplify formatting.

In addition, the initial values assumed by the formatter for line length, page length, etc. have been changed, to suit the printers in common use with the 380Z.

 Copyright May 1980

TXED VERSION 4 REFERENCE MANUAL

INDEX

380Z 2.1, C.2

ASCII **C.1**, 11.20, 11.22
Absolute 11.2
Advanced editing **12.4**
Algol 9.16
Ambiguous files 10.9, **10.10**
Anadex DP-8000 11.3, 11.4, 11.22, F.3
Argument **C.1**
Assembly Language 9.16
Asterisk 2.1, 2.3, 9.1, 9.2

B command **9.4**
BACKSPACE character **11.27**
Block commands **4.6**
Block copy **4.6**, 9.9
Block deletion **4.6**, 9.9
Block move **4.6**, 5.5, 9.9
Bold face 7.9, 9.12, **11.26**
Bootstrap **C.1**
Break 11.3, **11.13**
Buffer **C.1**

C 9.16
C command **9.5**
CAPS LOCK key **2.2**
CON: 10.1
COS Monitor **C.2**
CP/M 10.2, 10.4, 10.5, 10.6, 10.7, 10.10, 10.11, **C.2**
CTRL A command **8.1**
CTRL B command **4.3**, **8.1**
CTRL C command **5.4**, **8.1**, 9.4, 10.3
CTRL D command **2.6**, 4.3, **8.1**
CTRL E command **4.4**, **8.1**
CTRL F command **8.2**, 9.4
CTRL G command **2.6**, 2.8, 4.3, **8.2**
CTRL H 8.2
CTRL I **8.2**, 9.15
CTRL J command **4.5**, **8.2**
CTRL K command **4.4**, **8.2**
CTRL L **8.2**, 10.6
CTRL M 8.2
CTRL N command **2.6**, 2.8, 4.3, **8.2**, 12.1
CTRL O command **4.5**, **8.3**
CTRL P command **4.3**, 4.4, **8.3**
CTRL Q 12.4
CTRL Q command **4.3**, 4.4, **8.3**
CTRL R **9.4**
CTRL R command **4.4**, **8.3**
CTRL S command **2.7**, **8.3**

 Copyright May 1980

CTRL T 8.3
CTRL U **9.4**
CTRL U command **4.3**, **8.3**
CTRL V command **2.6**, 4.3, **8.3**, 12.1
CTRL W command **4.3**, **8.3**
CTRL X **9.4**
CTRL X command **8.3**, **12.1**, 12.2
CTRL Y command **4.5**, **8.3**
CTRL Z character **10.2**, 10.6, 9.12
CTRL Z command **4.3**, **8.3**
CTRL ↑ command 12.5, **8.3**
CTRL key **2.2**, **2.6**
Case shifting **2.7**
Centering **11.19**
Close **C.1**
Closing file 5.1, **10.2**
Command line **C.1**
Command mode **2.3**, 4.1, 8.4, **9.1**, **C.1**, 11.3, 12.1, 12.2
Command repetition 6.3
Command termination **9.1**
Command terminator **C.2**
Compound command 6.2, **9.3**
Conditional page **11.16**
Configuration E.1
Control character 2.2, 8.3, 9.7, **C.2**, 11.22
Corrections **2.5**
Cursor **2.5**, 9.16

D command **9.6**
DELT 2.5, 4.4, **9.4**
DELT key **2.2**, **2.6**
DIR command **3.3**
Default **C.2**
Deletion 2.5, **4.4**, **9.6**
Device name C.2, **10.1**
Directory 3.3, 5.2, **C.2**
Disc name **3.1**, **10.1**

EB command **5.2**, **10.2**, 10.3, 10.4, 10.5, 10.6, 10.7, 10.11, 10.12, 11.24
ED command **5.2**, **9.14**, 10.10
EF command 10.2, **10.5**, 10.6, 10.8, 11.3
EI command **10.4**, 10.15
EL command **4.2**, **9.14**
EM command 9.13, **9.14**, 12.3
ER command **5.5**, **10.4**, 10.6, 10.8
ES command **9.16**
ESC character 11.27, 12.2, 12.5
ESC key **2.2**, 2.3, 3.3, 9.1
ET command 9.10, **9.15**, 11.13
EW command **5.5**, 10.2, **10.5**, 10.8, 10.10, 10.15, 11.3
EX command **3.3**, 9.17, 10.2, **10.3**, 10.4, 10.5, 10.7
Edit backup **3.5**, **5.3**, **C.3**, 10.2, 10.3
Edit file **3.3**, 5.2
Editing **2.5**
Editing macros **12.5**
Equals sign 9.2

Erase file 5.2
Error message **C.3**
Error messages **B.1**
Exclamation mark 2.3
Extension 5.2, **C.3**

F command **6.1**, **9.8**
FORM FEED **8.2**, 10.6, 11.7
File **C.3**
File directory 5.2
File name **3.2**, **C.3**, **10.1**, 10.2, 10.3
File specification **3.1**, **C.3**, **10.1**, 10.2, 10.11
File type **3.2**, **C.3**, **10.1**, 10.2, 10.3, 10.11
Files **3.1**, 5.1, **10.1**
Fill and justify **11.2**, **11.11**, 11.12
Filling **7.5**, 11.1
Finding character **4.5**
Finding marker **4.5**
Folio 7.9, **C.3**, **11.8**
Formatted printing **7.4**
Formatter 9.17
Formatting 1.1, **7.1**, 9.11, **C.4**, **11.1**
Formatting command **7.5**, 7.9, 9.10, **C.4**, **11.2**
Formatting long files **11.23**
Front Panel **8.2**
Full indentation **11.19**

G command 9.12, **9.13**, 9.14
General advice **3.5**

Hanging indent **11.18**
Heading **11.8**
Horizontal scrolling **9.15**
How it works **3.4**, **5.3**

I command **9.7**
Immediate mode **2.3**, 4.1, **8.1**, **C.4**, 12.1, 12.2
Indentation **11.17**
Input file **C.4**
Inserting marker **4.5**
Insertion 2.5, 4.4
Insertion of text **9.7**
Iteration **9.3**, **C.4**, 12.3
Iteration brackets **9.3**

Justification **7.5**, 11.1, **11.11**

K command **9.6**, 10.8
Keyboard **2.2**

L command **9.5**, 10.8
LINE FEED character 9.10
LINE FEED key 4.5
LP: 10.1
LST: 10.1
Large Files **5.6**

 Copyright May 1980

Layout commands **11.11**
Leaving space for diagram **11.22**
Line **C.4**
Line spacing **11.13**, 11.14
Listing **C.4**
Literal 11.1
Literal mode 11.3, **11.12**, 11.14, 11.15
Logged in disc **3.2**, **C.4**
Long lines **2.5**, **4.1**, 9.15

M command **9.13**, **12.1**
Macro command 6.3, 8.3, **9.13**, 9.14, **C.5**, **12.1**, 12.3, 12.4
Macro/Save buffer 9.12, 9.13, 9.14, 9.18, **C.5**, 10.8, 12.3
Merging files **5.4**
Modification of text **9.7**
More on input **5.6**, 8.2

N command 5.6, 10.2, **10.3**, 10.4, **10.7**, 11.24
No room **5.6**
Numeric argument 6.1, **9.1**, 11.2

O command **9.7**
OKI Microline 80 F.3, 11.3, 11.4, 11.27
Open **C.5**
Opening file 5.1, **10.2**
Operating System 3.3, 5.4, 9.17, 10.3, 10.4
Output file **C.5**
Overprinting **11.27**

P command **7.4**, 9.10, **9.11**, 12.4
PIP **C.5**
PUN: 10.1
Pad character 11.18, 11.19, **11.20**, 11.21
Page limits **11.4**, **11.7**, **11.10**
Page of input file **C.5**
Pointer movement **4.3**, 9.4
Pound sign **11.27**
Printer 9.12, F.1, 11.3, 11.22
Printer option 7.1, 9.10, F.1
Printing **7.4**, 9.10, 11.3
Prompt 9.1, **C.5**

Q* command **9.17**, 9.18
QC command **4.6**, **9.9**, 10.8
QD commadn **9.9**
QE command **5.2**, 10.5, **10.9**
QF command **9.16**
QK command **4.6**, **9.9**
QL command **5.2**, **10.9**
QM command **4.6**, **9.9**, 12.6
QP command 7.9, 9.10, **9.11**, 9.17, 10.8, 10.14, 11.3, 11.23, 11.24
QR command **5.2**, 10.5, **10.9**
QS command **4.1**, **9.16**, 9.17
QT command **7.5**, 7.9, 9.10, **9.11**, 9.12, 9.17, 10.8, 10.14, 11.3, 11.23, 11.24
QU command **9.17**
QV command 7.9, 9.10, 9.12, **9.12**

QW command 7.9, 9.10, 9.12, 9.17, 10.2, 10.6, **10.8**, 10.15, 11.3, 11.23, 11.24, 12.6
QX command **5.4**, **9.17**, 10.4, 10.12
QZ command **9.12**
Quick Reference Guide A.1
Qume Sprint 5 F.4, 11.4, **11.25**, 11.28

R command **5.5**, 10.2, 10.4, **10.6**, 10.7, 11.18, 11.24
RDR: 10.1
REPT key **2.2**
RETURN **9.4**
RETURN character 9.17, 11.3
RETURN key **2.2**, 2.3, 2.5, 8.2
Ragged right **11.2**, **11.11**, 11.12
Reading **C.5**, **10.2**
Reading from files **3.1**
Relative 11.2
Rename file 5.2
Repetition 6.3
Restart address **C.5**
Right margin 4.1, 9.17

S command **9.12**, 9.14, 10.8
SHIFT key **2.2**
SPACE **9.4**
STAT **C.6**
Saving text **9.12**
Selective output **11.24**
Simple command **9.3**
Standard letter **12.5**
Starting a new line **11.13**
Starting a new page **11.16**
String replacement **6.2**, **9.8**
String search **6.1**, **9.8**, 12.2
String substitution **6.2**
Subscript **11.27**
Superscript **11.27**

T command 9.10, **9.11**
TAB character **8.2**, 9.10, 9.15, **C.6**, 11.12, 11.13, 11.15
TAB stop 9.15
TEX G.1, 11.1
TTY: 10.1
TXED fsp **3.2**, 3.3, 5.1
Text buffer **2.7**, **C.6**
Text markers **4.5**, **9.8**, **C.6**, 11.24, 12.4, 12.6
Text pointer **2.7**, **C.6**
Text string **C.6**
Text string argument **9.2**
Text window **2.8**, 4.2, 5.2, 9.11, **C.6**, 10.10
Two column listing **12.4**

U command **9.5**
Underlining 7.9, 9.12, **11.25**
Underlining convention 1.2

Version number **2.1**

W command 10.2, **10.7**, 10.8, 10.13
Working copy 1.2, D.1
Writing **C.6**, **10.2**
Writing to files **3.1**

X command **6.2**, **9.8**

Y command **9.6**

Z command **9.5**

↑ command **9.7**

.BC command 11.26
.BE command 10.14, 10.15, **11.23**, 11.25
.BP command 11.8, **11.16**, 11.17, 11.23
.BR command 11.3, **11.13**
.BS command 11.4, 11.6, **11.7**, 11.10
.CC command **11.22**
.CE command **7.6**, 11.2, **11.19**, 11.20
.CI command **7.6**, **11.18**
.CP command **11.16**
.DS command **11.13**, 11.14, 11.15
.E1 command **11.27**
.E2 command **11.27**
.E3 command **11.27**
.E4 command **11.28**
.E5 command **11.27**, 11.28
.EN command 10.14, 10.15, 11.23, **11.24**
.FJ command **7.6**, **11.11**, 11.12
.FS command 11.9, **11.10**
.HS command 11.8, **11.9**
.IF command **11.20**
.IG command **11.21**, 11.22
.IN command **7.6**, 11.6, 11.13, **11.17**, 11.18, 11.19, 11.22
.LI command **7.6**, **11.12**, 11.13
.LL command 11.4, **11.6**, 11.11, 11.17, 11.22
.LM command 9.12, 11.4, **11.6**, 11.13, 11.17
.LS command **11.13**, 11.14, 11.15
.MF command 11.7, 11.8, **11.9**, 11.10, 11.11
.MH command 11.7, **11.8**, 11.9, 11.10, 11.11
.NC command **11.10**, 11.25
.OP command **11.8**, 11.10, 11.11
.PA command **11.16**, 11.17, 11.23
.PC command 11.2, **11.20**, 11.23, 11.25
.PL command 11.4, **11.6**, 11.7, 12.6
.PN command **11.8**
.PP command **7.6**, 11.12, **11.15**
.QN command 9.12, **11.25**
.QY command **11.25**
.RR command **11.11**, 11.12
.SP command 11.2, 11.12, **11.14**, 11.15, 11.19, 11.22, 11.23
.SS command 11.12, **11.13**, 11.14, 11.15
.ST command **11.22**

 Copyright May 1980

.TI command 11.2, 11.15, **11.17**, 11.18, 11.19, 11.21
.TS command 11.4, 11.6, **11.7**
.UC command **11.25**

THIS PAGE INTENTIONALLY LEFT BLANK

 Copyright May 1980